ECB模式利用与CBC翻转攻击

原创

delta_hell 于 2020-02-29 16:35:45 发布 631 收藏

分类专栏： 安全

本文链接：https://blog.csdn.net/u011317663/article/details/104576787

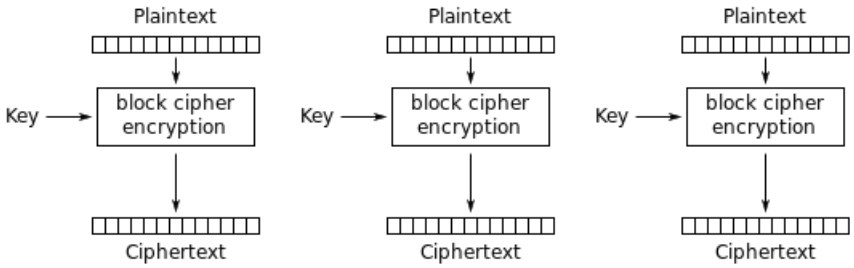
ECB模式利用与CBC翻转攻击

前言

最近看WEB安全，其中讲到cookie操纵的时候， 提到ECB与CBC加密模式的利用，实际操作了一下，记录一下。

ECB缺陷

ECB加密过程:



Electronic Codebook (ECB) mode encryption

从图中可以看出加密过程，是明文分组后，**每个分组被分别加密**，所以明文相同的两个分组，被加密出来的密文分组也是一样的。换句话说，这种加密方式，没法完全隐藏明文信息。

常见的一种利用方式，就是构造分组，获取想要的明文对应的密文。

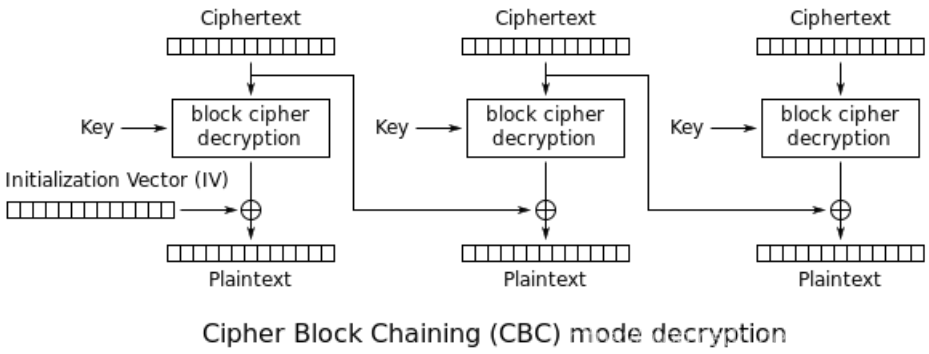
一种典型的场景，假如某系统用户名使用ECB模式加密，我们想获取用户A（假设A对应一个分组，简化理解，多个分组同理）的权限，就得设法构造A加密后的密文，但是我们又无法再次注册一个A用户。这种情况下，我们注册一个(PAD)A的用户,(PAD)就是一个分组填充，如8字节或16字节，依据加密算法而定，具体需要尝试。这样就会得到(PAD)A的密文，而该密文对应的最后一个分组，就是A对应的密文。

```
使用AES ECB 16字节分组，KEY=“8765432187654321”字节一个分组加密，测试结果如下：
明文：admin
密文：B9CB9D5D68EF28B73A1404AD23B15C03
明文：xxxxxxxxxxxxxxxxadmin
密文：176EF38A69F2AF33430D9EFD39ED5737【B9CB9D5D68EF28B73A1404AD23B15C03】
(【】为标注使用，并非密文结果)
```

从上面测试结果可以看出，最后一个分组(16字节)完全一致。

CBC翻转攻击

CBC的解密过程：



Cipher Block Chaining (CBC) mode decryption

从解密过程可以看出，CBC的解密依赖前一个分组的密文作为IV与中间结果异或， 这个增加了复杂度，但是从另一个方面也可以看出，也可以通过修改前一个密文值来得到想要的明文。原理即是利用异或的特性，两个相同值异或，结果为0。

异或特性：0 ^ b = b ; a ^ a = 0 ; a ^ a ^ b = b;

假设当前CBC分组为第i个分组，则cipher为对应的分组密文，plain为明文，inter为中间值, 以修改第i个分组第j个字节为pseduo_char为例。则过程如下：

```
【1】inter[i] = cipher[i] decrypt

【2】plain[i] = inter[i] ^ cipher[i-1]

【3】plain[i][j] = inter[i][j] ^ cipher[i-1][j]  #第j个字节

【4】0 = inter[i][j] ^ cipher[i-1][j] ^ plain[i][j]

【5】pseudo_char = inter[i][j] ^ cipher[i-1][j] ^ plain[i][j] ^ pseudo_char
```

所以，若修改前一个密文分组第j个字节：

pseudo_cipher[i-1][j] = cipher[i-1][j] ^ plain[i][j] ^ pseudo_char

则由式【3】【5】可以得出

plain[i][j] = inter[i][j] ^ pseudo_cipher[i-1][j] = pseudo_char

即可以得到想要的值。

根据上面的推导，对多个字节可以使用同样的方式进行修改。若是第0个分组，则cipher替换为iv即可。(大部分使用场景中，iv是附加在密文前面传输的)

```
使用DES CBC 8字节，KEY="87654321"，IV="87654321"，试验从xxxxxxxxadmix得到xxxxxxxxadmin.过程如下：
明文：xxxxxxxxadmin
密文:919B08D902104B8809620E369EC30477
明文：xxxxxxxxadmix
密文:919B08D902104B88899829F860AF29EA
修改后的密文:919B08D914104B88899829F860AF29EA
解密后的明文：+~@2Padmin
修改后的IV:6B310E7F5C494119
再次解密后的明文：xxxxxxxxadmin
```

该使用场景也适用于上节ECB的场景，操纵cookie越权之类的。

因为该方式是以损坏前一个分组密文为前提，所以存在伪造的密文在解密过程中出现pad error或其它不能解密的情况。

总结

从上面的试验可以看出，ECB貌似很不安全，即使不用上面说的方式来做，通过多份数据比较，可能也能获取一定的信息；而CBC好像问题也挺多，本文讲的是构造密文得到想要的明文，而之前写的一篇Padding Oracle则可以在不需要秘钥的情况下获取明文，感觉也是筛子。。。当然，这些都是实验场景，实际运用肯定要复杂的多，比如确定加密方式(最近看一道CTF的加密题有感)。。。

附录:

1. Block cipher mode of operation
2. CBC字节翻转攻击