CS2107 Live Class Lec1

$_/|/|U_Ch@NgRu!$

May 11, 2021

1 Oracle in security analysis

1.1 Oracle definition

An oracle is a person or agency considered to provide wise and insightful counsel or prophetic predictions, most notably including precognition of the future, inspired by deities.

1.2 Oracle types

1. Encryption oracle: on query a plaintext x, the oracle outputs the ciphertext $E_k(x)$ where the key k is a secret key
2. Decryption Oracle: on query a ciphertext c, the oracle outputs the plaintext D_k(x) where the key k is a secret key

1.3 Padding Oracle attack

The attacker have:

1. A ciphertext which include the iv:(iv,c)
2. Access to the padding oracle

The attacker's goal: the plaintext of (iv,c)
Note: the ciphertext is encrypted with a secret key k. The padding oracle knows the key

1.3.1 The step flow

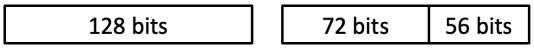
1. The attack: send a ciphertext to Oracle
2. The oracle:

(a) Yes, if the plaintext is in the correct ”padding” format

(b) No, other wise

1.3.2 The padding format

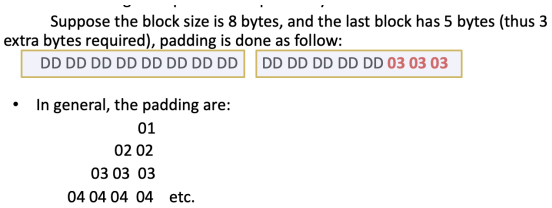
The block size of AES is 128 bits(or 16 bytes). Suppose the length of the plaintext is 200 bits, it will be fitted into two blocks, with the remaining 56 bits "padded" with some values



Manys to fill the padding bits, but the number of padding bits should be recorded or stored in the plaintext to guarantee the integrity of the plaintext
A common standard is PKCS#7

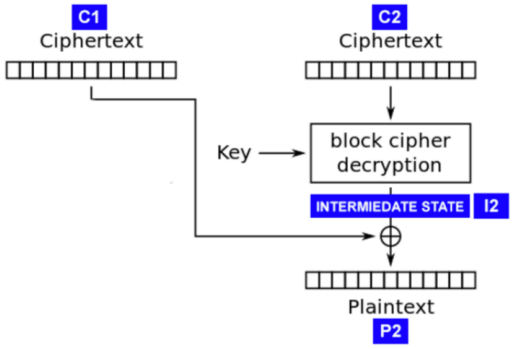
1.3.3 PKC#7

Padding is in whole bytes. The value of each added byte is the number of bytes that are added, i.e. N bytes, each of value N are added. The number of bytes added will depend on the block boundary to which the message needs to be extended.



1.3.4 Padding oracle attack on AES CBC mode

AES CBC mode is not secure against padding oracle attack(when padding is done with PKCS#7)
Chosen ciphertext attack can be applied here



The image above shows the decryption process of CBC, by brute forcing the last bit of C1, the last bit of Intermiedate state I2 can be found. Because once the oracle accept, it means the modified last character of C1: cLast⊕the last character of I2=0x01 After get the last bit of I2, we can extends to brute force the last two bits of I2
Reference:https://jiang-zhenghong.github.io/blogs/PaddingOracle.html
Note that the attack is practical because there are protocols(which are interactions between two or more entities) between a client and server which performs this

2 Cryptography pitfall

2.1 Wrong choice of IV and reusing one-time-pad

2.1.1 Reuse of IV

Some application applies wrong IV generation mechanism where the IV may be reused(E.g Some implementation applies the file name as the IV while it is quite common to have files with the same filename)(e.g. Microsoft R4, flow, where when user modify the file, both the IV and key are not refreshed and a replay attack can be made)
Reference: https://www.schneier.com/blog/archives/2005/01/microsoft_rc4_f.html http://eprint.iacr.org/2005/00

2.1.2 IV is predictable

BEAST attack (Browser Exploit Against SSL/TLS.)
The TLS protocol uses symmetric encryption with block ciphers
If the same data and the same key always gave the same encrypted content, an attacker could easily break any encryption. That is why TLS uses initialization vectors. This means, that encryption is seeded using random content.
the attacker only needs to guess a single byte at a time. This can be done if the attacker can predict most of the data (for example, HTML code) and needs just one piece of secret information, for example, a password. The attacker can then test the encryption carefully, selecting the right length of the data, so that they have just one byte of information in a block that they do not know. And then, they can test the block just for 256 combinations of this byte. Then, they repeat the process for the next byte, soon coming up with the entire password.(The IV is predicatable because most web side start the same)

2.1.3 reusing one-time-pad

Verona project

2.2 Predictable generation of secret

In C

```
#include<time.h>
#include<stdlib.h>
srand(time(NULL));
int r=rand()
```

If the above code is used to generate a random IV or even password, the attacker:

1. If the attacker knows when the IV and password is generated, he can know the IV and password
2. If the attacker knows the approximate time, he can brute force it
3. even if the attack does not know any information about the time, he can also brute force the variable s. This is because int data type in C is only either 2-byte(16 bit) or 4 byte(32 bit)

Linux (with -lbsd) now all have an implementation of arc4random that is crypto-secure and that cannot fail. That makes it a very attractive option:

```
char myRandomData[50];
arc4random_buf(myRandomData, sizeof myRandomData); // done!
```

The other way: you can use the random devices as if they were files

```
int byte_count = 64; char data[64];
FILE *fp;
fp = fopen("/dev/urandom", "r");
fread(&data, 1, byte_count, fp);
fclose(fp);
```


3 Kercknoffs's principle

A system should be secure even if everything about the system, except the secret key, is public knowledge