The RC4 encryption

The RC4 encryption algorithm is a symmetric encryption algorithm.

Symmetric encryption algorithm

Symmetric encryption (also called private key encryption) refers to an encryption algorithm that uses the same key for encryption and decryption . Sometimes called the traditional cryptographic algorithm, the encryption key can be calculated from the decryption key, and the decryption key can also be calculated from the encryption key. In most symmetric algorithms, the encryption key and the decryption key are the same, so this encryption algorithm is also called a secret key algorithm or a single key algorithm. It requires the sender and receiver to agree on a key before communicating securely. The security of the symmetric algorithm depends on the key. Leaking the key means that anyone can decrypt the messages they send or receive, so the confidentiality of the key is very important to the security of communication.

RC4 encryption algorithm

RC4 algorithm is characterized by simple algorithm, fast running speed, and the key length is variable, the variable range is 1-256 bytes (8-2048 bits), under the premise of technical support today, when the key length is At 128 bits, it is not feasible to search for keys by brute force, so it can be predicted that the key range of RC4 can still resist brute force search key attacks for a long time in the future. In fact, no effective attack method has been found for the 128bit key length RC4 encryption algorithm.

The principle of RC4 algorithm is very simple, including initialization algorithm (KSA) and pseudo-random sub-code generation algorithm (PRGA) two parts. Assume that the length of the S-box is 256 and the key length is Len.
Let's take a look at the initialization part of the algorithm (in C code):
Among the:
Parameter 1 is a 256-length char type array, defined as: unsigned char sBox[256];
Parameter 2 is the key, and its content can be defined arbitrarily: char key[256];
Parameter 3 is the length of the key, Len = strlen(key);

```
/* Initialization function */
void rc4_init(unsigned char *s,unsigned char *key, unsigned long Len)
{
    int i = 0 ,j = 0 ;
```

```
     char k[ 256 ]={ 0 };
    unsigned char tmp = 0 ;
     for (i = 0 ;i< 256 ;i++ ) {
         s[i] = i;
         k[i] =key[i% Len];
    }
    for (i= 0 ;i< 256 ;i++ ) {
         j =(j+s[i]+k[i])% 256 ;
         tmp = s[i];
         s[i] = s[j]; // Exchange s[i] and s[j]
         s[j]= tmp;
    }
}
```

During the initialization process, the main function of the key is to scramble the S-box, i ensure that each element of the S-box is processed, and j ensures that the scramble of the S-box is random. Different S-boxes can obtain different sub-key sequences after being processed by the pseudo-random sub-cipher generation algorithm. Perform an xor operation on the S-box and the plaintext to obtain the ciphertext, and the decryption process is exactly the same.
Let's take a look at the encryption part of the algorithm (in C code):
Among them.
Parameter 1 is the disturbed S-box in the rc4_init function above;
Parameter 2 is the data data that needs to be encrypted;
Parameter 3 is the length of data.

```
/* Encryption and decryption */
void rc4_crypt(unsigned char *s,unsigned char *Data,unsigned long Len)
{
    int i = 0 , j = 0 , t = 0 ;
    unsigned long k = 0 ;
    unsigned char tmp;
     for (k = 0 ;k<Len;k++ )
    {
         i = (i + 1 )% 256 ;
         j =(j+s[i])% 256 ;
         tmp = s[i];
         s[i] = s[j]; // Swap s[x] and s[y]
         s[j]= tmp;
         t =(s[i]+s[j])% 256 ;
         Data[k] ^= s[t];
    }
```

```
}
```

Let's take a look at the complete program

```
// Program start
#include<stdio.h>
#include < string .h>
typedef unsigned longULONG;

/* Initialization function */
void rc4_init(unsigned char *s, unsigned char *key, unsigned long Len)
{
    int i = 0 , j = 0 ;
     char k[ 256 ] = { 0 };
    unsigned char tmp = 0 ;
     for (i = 0 ; i< 256 ; i++ )
    {
        s[i] = i;
        k[i] = key[i% Len];
    }
    for (i = 0 ; i< 256 ; i++ )
    {
        j = (j + s[i] + k[i])% 256 ;
        tmp = s[i];
        s[i] = s[j]; // Swap s[i] and s[j]
        s[j] = tmp;
    }
}

/* Encryption and decryption */
void rc4_crypt(unsigned char *s, unsigned char *Data, unsigned long Len)
{
    int i = 0 , j = 0 , t = 0 ;
    unsigned long k = 0 ;
    unsigned char tmp;
     for (k = 0 ; k<Len; k++ )
    {
        i = (i + 1 )% 256 ;
        j = (j + s[i])% 256 ;
        tmp = s[i];
```

```
        s[i] = s[j]; // Swap s[x] and s[y]
        s[j] = tmp;
        t = (s[i] + s[j])% 256 ;
        Data[k] ^= s[t];
    }
}

int main()
{
    unsigned char s[ 256 ] = { 0 }, s2[ 256 ] = { 0 }; // S-box
    char key[ 256 ] = { " justfortest " };
     char pData[ 512 ] = " This is an encryption Data " ;
    unsigned long len = strlen(pData);
     int i;

    printf( " pData=%s\n " , pData);
    printf( " key=%s,length=%d\n\n " , key, strlen(key));
    rc4_init(s, (unsigned char *)key, strlen(key)); // Has completed the initialization
    printf( " Complete the initialization of S[i], as follows: \n\n " );
     for (i = 0 ; i< 256 ; i++ )
    {
        printf( " %02X " , s[i]);
         if (i && (i + 1 )% 16 == 0 )putchar( ' \n ' );
    }
    printf( " \n\n " );
     for (i = 0 ; i< 256 ; i++) // Use s2[i] to temporarily reserve the initialized s[i], it
is very important! ! !
    {
        s2[i] = s[i];
    }
    printf( " Initialized, now encrypted:\n\n " );
    rc4_crypt(s, (unsigned char *)pData, len); // Encryption
    printf( " pData=%s\n\n " , pData);
    printf( " Already encrypted, now decrypt:\n\n " );
     // rc4_init(s,(unsignedchar*)key,strlen(key)); // initialize the key
    rc4_crypt(s2, (unsigned char *)pData, len); // Decrypt
    printf( " pData=%s\n\n " , pData);
     return  0 ;
}
```

```
//The program is over
```

By Boschko

- My Hack The Box: https://www.hackthebox.eu/home/users/profile/37879
- My Website: https://olivierlaflamme.github.io/
- My GitHub: https://github.com/OlivierLaflamme
- My WeChat QR below:

Revision #2
Created 5 September 2020 21:31:32 by Boschko
Updated 5 September 2020 21:55:11 by Boschko