CRC32的逆向分析与源码

原创

瞧红尘 于 2013-10-28 00:16:51 发布 2825 收藏
分类专栏： PEDIY 文章标签： CRC32 算法 逆向 破解

文章首发于看雪(http://bbs.pediy.com/showthread.php?t=120018),现在整理移到这里.

```
' ******************** CRC验证 ********************
'返回验证码         二进制 串
Public Function CRC32(ByRef bArrayIn() As Byte, ByVal lLen As Long) As Long
    Dim lCurPos As Long
    Dim lTemp As Long
    Dim crcTable(0 To 255) As Long
    Dim i As Long, x As Long, crc As Long
    Const Limit = &HEDB88320

    For i = 0 To 255
        crc = i
        For x = 0 To 7
            If crc And 1 Then
                crc = (((crc And &HFFFFFFFE) \ 2) And &H7FFFFFFF) Xor Limit
            Else
                crc = ((crc And &HFFFFFFFE) \ 2) And &H7FFFFFFF
            End If
        Next x
        crcTable(i) = crc
    Next i
    '以上初始化 出来一个表(是固定的)
    If lLen < 0 Then Exit Function
    lTemp = &HFFFFFFFF

    For lCurPos = 0 To lLen
        lTemp = (((lTemp And &HFFFFFF00) \ &H100) And &HFFFFFF) Xor (crcTable((lTemp And 255) Xor bArrayIn(
      '前三位*3 xor 表(第四位*1 xor 新字节*1)*4
    Next lCurPos

    '取反输出结果 -->结果取其十六进制标记就是我们常见的CRC32了
    CRC32 = lTemp Xor &HFFFFFFFF
End Function
```

关键部分就再循环,他做了什么呢
它每次加入一个新字节,和先前计算的结果4字节进行计算

先前由(高到底): A1 A2 A3 A4

新加入的字节:　　B(n)
计算结果　:　C1 C2 C3 C4

计算方式:(注意 ^ 为 异或计算)

```
┌→A1 A2 A3 A4 →A4 ^ B(n)
│  ↘ ↘ ↘    ↘ ↙
│ 00 A1 A2 A3    X
│ ^ ^ ^ ^   ↓(查表)
│ T1 T2 T3 T4 ←←T(X)
│ ↓↓↓↓
└─T1 C2 C4 C4
  ↓↓↓↓(最后一次完成后取反)
  R1 R2 R3 R4      ;CRC32 结果
```

新值C经过一次计算后,他的最高位C1没变,是查表的最高位T1
那么这个表T1的值是不是唯一的呢,我们把表打印出来(见文章末尾),看一下
可喜的是,最高位(00-FF)都是一对一得关系,从00到FF没有重复,那么逆向推导就成为可能了

现在我们知道了文件得CRC32,
　那么就知道最后一次循环后得值(取反可得),这个值得最高位,就是它最后一次查表的值
　由于这个表和查表是一一对应关系,我们可以知道两点
1.　T1 可以得到 T(X) 也就是 T1 T2 T3 T4
2.　T1 可以得到 X　也就是 A4 ^ B(n)

由 T(X) 和 C 我们可以得到 A1 A2 A3 ,这是倒数第一次循环计算的高位3字节,
用同样的方法我们可以得到
　倒数第二次循环计算的高位2字节
　倒数第三次循环计算的高位1字节
　倒数第四次循环计算的高位0字节 (也就是说4次这样的循环计算后,再前面的结果就是任意的了)

一次循环,会模糊掉一个字节位的crc32校验,4次循环会模糊掉四字节的CRC32校验

这样只要指定最后4个字节,我们就可以达到修改出任意CRC32

现在,我们要把一个CRC校验文件添加4字节
CE037B28 转换为 09635B62
　(这也是我最初的实验，很痛苦；计算试了几次没成功，从算法上看又可行，因为自己一没注意就计算错了)

```
CE037B28 -> 09635B62  ;目的CRC
'取反
31FC84D7 -> F69CA49D
        ↓↓↓↓
    ↓9CA49D
        F6B9265B   D5 {表} F6查表→D5（X的值）
        ↓↓↓↓  ↓      →F6B9265B（T的值）
    F62582C6  ↓
去掉最高位  F6       ↓
         ↙↙↙  ↙ ; 向高位移动
        2582C6[*D5]   ; N1 xor old1 =x1
```

2582C6[^D5];
  82C6 [^D5]
25 6FD2 A0 B2 {表}

25 ED14 [^D5^A0]

ED14 [^D5^A0] [^B2]

EDB88320 =80 {表}

 AC[^D5^A0^83][^A5^20][^6D]

 AC BC F9 40 2B {表}
[^D5^A0^83^BC][^B2^20^F9][^80^40][^2B]

31 FC 84 D7 ；与原来的31FC84D7再次异或
7B 97 44 FC ；这是最后添加的4个字节

为什么到最后要这样再异或一次呢，
  因为A^B=C → A^C=B → B^C=A
原来CRC取反 A1 A2 A3 A4 A
修改4字节CRC取反 B1 B2 B3 B4 B
目的CRC取反 C1 C2 C3 C4 C
  要的到A^B=C 我们已知A，C 异或一下就得到我们需要的B了，虽然还需要经过一段运算

认真的读者会发现，CRC32逆运算更容易得到想要的CRC的值，而正运算几乎是不可能的，
因为我们每加一新数据时候，只能确定CRC4字节中的1字节，而其他字节都会有影响的。
==========

问题二：
如何在文件中任意部分覆盖4字节得到想要的CRC值呢，假定一个文件长度为len，覆盖位置offset，文件最后CRC结果为d（取反结果为D）

首先我们想，如果覆盖掉最后4字节的话，
 我们只需要先计算文件前面 len-4 字节CRC，再添加4字节就行了，这个很简单。

如果在文件中央，
我们要使文件最后的CRC满足条件D，而我们可以控制的地方只有（offset~offset+3）4字节区域，
分成3部分：
前部分，覆盖部分（4字节），后面部分

  前部分：CRC可以计算出来
  后面部分，如果知道最终的目标CRC，又知道它每一次计算的B（n），
那么倒数任何次的A1 A2 A3 A4 ，T1 T2 T3 T4，X，C1 C2 C3 C4都可以计算出来的，
计算方法和我们上面分析的 X 是不可以确定的，用他 X ^ B（n） 就得出A4了

  那么我们 覆盖掉这4字节的任务就是这样了，后面部分要达到CRC结果为D的，那到覆盖部分的尾端时候需要的为C，
  而前面覆盖部分可以直接计算得到结果A，覆盖部分只要把A调整为C（而不是直接调整为D）

思路好了，我附上原代码吧，知道大家都喜欢这个

VB代码

```
'**************************************************************************
'**模 块 名：mCRC
'**说    明：E_mail:xa04@qq.com 版权所有2008 - 2009(C)1
'**创 建 人：瞧红尘
'**日    期：2010-09-03 02:19:08
'**修 改 人：
'**日    期：
'**描    述：
'**版    本：V1.0.0
'**************************************************************************
Option Explicit


Private Declare Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" (Destination As Any, Source As Any,

Private Type t_CRCdata
    num1 As Byte
    num2 As Byte
    num3 As Byte
    num4 As Byte
End Type

' ******************** CRC32编码 与 编辑 ********************
'第一参数传入字节数组，
'第二个写偏移覆盖（覆盖掉4字节），
'第三个写你想反回个什么样的crc，
'四个就是你的返回值
Public Function CRC32Edit(pByte() As Byte, _
                        Optional lOffset As Long = -1, _
                        Optional lCRC32 As Long, _
                        Optional retlCRC As Long) As String

    Dim crc32Result As Long
    Dim i           As Long
    Dim j           As Long
    Dim dwCrc       As Long
    Dim iLookup     As Long
    Dim Lb          As Long
    Dim Ub          As Long

    Dim Buffer()    As Byte
    Dim Buffer2()   As Byte
    Dim k           As Long
    Dim cr1         As t_CRCdata
    Dim cr2         As t_CRCdata

    '常数
    Const Num0           As Long = &H0
    Const Num1           As Long = &H1
    Const Num2           As Long = &H2
    Const Num8           As Long = &H8
    Const Num255         As Long = &HFF
    Const Num256         As Long = &H100
```

```
Const Num16777215    As Long = &HFFFFFF
Const dwPolynomial   As Long = &HEDB88320
Const Num2147483647  As Long = &H7FFFFFFF
Const NumNegative1   As Long = &HFFFFFFFF
Const NumNegative2   As Long = &HFFFFFFFE
Const NumNegative256 As Long = &HFFFFFF00

'CRC32表
Dim crc32Table(&HFF) As Long
Dim CRCdata(255) As t_CRCdata

'初始化CRC32表*****************************
For i = Num0 To Num255
    dwCrc = i
    For j = Num8 To Num1 Step NumNegative1
        If (dwCrc And Num1) Then
            dwCrc = ((dwCrc And NumNegative2) \ Num2) And Num2147483647
            dwCrc = dwCrc Xor dwPolynomial
        Else
            dwCrc = ((dwCrc And NumNegative2) \ Num2) And Num2147483647
        End If
    Next
    crc32Table(i) = dwCrc
    CopyMemory CRCdata(i), crc32Table(i)
Next
crc32Result = NumNegative1  '初始化

If lOffset < 0 Then
    lOffset = UBound(pByte) + 1
Else
    If lOffset > UBound(pByte) Then
        lOffset = UBound(pByte) + 1
    End If
End If

If lOffset > 0 Then
    ReDim Buffer(lOffset - 1)

    CopyMemory Buffer(0), pByte(0), lOffset   '数据拷贝
    'Debug.Print "lOffset:", lOffset

    '计算CRC32码*******************************
    Lb = LBound(Buffer)
    Ub = UBound(Buffer)
    For i = Lb To Ub
        iLookup = (crc32Result And Num255) Xor Buffer(i)  '第四位 xor 新字节
        crc32Result = ((crc32Result And NumNegative256) \ Num256) And Num16777215 '前三位
        crc32Result = crc32Result Xor crc32Table(iLookup) '前三位*3 xor 表(第四位*1 xor 新字节*1)*4
    Next i
End If

CRC32Edit = Hex$(Not (crc32Result))    '计算前面的crc值,返回是什么样的

'计算后面的
k = UBound(pByte) - lOffset - 3
If k > 0 Then   '后面是否有数据
    ReDim Buffer2(k - 1)
    CopyMemory Buffer2(0), pByte(lOffset + 4), k
    k = Not (lCRC32)
    CopyMemory cp1, k
```

```
        CopyMemory cr1, k
        '反向CRC计算

        For j = UBound(Buffer2) To 0 Step -1
            For i = 0 To 255
                If cr1.num4 = CRCdata(i).num4 Then
                    Exit For
                End If
            Next
            If i > 255 Then
                Debug.Assert False
            End If

            cr1.num4 = cr1.num3 Xor CRCdata(i).num3
            cr1.num3 = cr1.Num2 Xor CRCdata(i).Num2
            cr1.Num2 = cr1.Num1 Xor CRCdata(i).Num1
            cr1.Num1 = i Xor Buffer2(j)
        Next
    Else
        k = Not (lCRC32)
        CopyMemory cr1, k
    End If
    '要得出文件CRC值=lCRC32  之前部分的CRC必须满足的条件:cr1的值

    '计算
    For j = 0 To 3
        For i = 0 To 255
            If cr1.num4 = CRCdata(i).num4 Then
                Exit For
            End If
        Next
'        If i > 255 Then
'            Debug.Assert False
'        End If

        cr1.num4 = cr1.num3 Xor CRCdata(i).num3
        cr1.num3 = cr1.Num2 Xor CRCdata(i).Num2
        cr1.Num2 = cr1.Num1 Xor CRCdata(i).Num1
        cr1.Num1 = i
    Next

    CopyMemory k, cr1
    k = k Xor crc32Result

    retlCRC = k '计算出来的是4字节覆盖的值,不是crc
End Function
```

下面是c代码

```
int CRC32Edit(char *byt,int bytLen,int lOffset, int lcrc,int &retlCRC)
{
 #define Limit   0xEDB88320
    #define Num0                            0
    #define num1                            0x1
    #define num2                            0x2
    #define Num8                            0x8
    #define Num255                          0xFF
```

```
   #define Num256                          0x100
   #define Num16777215                     0xFFFFFF
   #define dwPolynomial                    0xEDB88320
   #define Num2147483647                   0x7FFFFFFF
   #define NumNegative1                    0xFFFFFFFF
   #define NumNegative2                    0xFFFFFFFE
   #define NumNegative256                  0xFFFFFF00

int CRCdata[256];

bytLen--;
int i,x,crc;
int crcTable[256];
for(i=0;i<=255;i++)
{
 for(crc=i,x=0;x<8;x++)
 {
  if(crc & 1)
   crc=(((crc & 0xFFFFFFFE) / 2) & 0x7FFFFFFF) ^ Limit;
  else
   crc=(crc & 0xFFFFFFFE) / 2;
 }
 crcTable[i] = crc;
 CRCdata[i]= crc;
}
if(bytLen<0)return 0;
DWORD crcResult = NumNegative1;
if (lOffset<0 || lOffset>bytLen)
 lOffset=bytLen+1;

for(i = 0;i<= lOffset-1;i++)
{
 crcResult=(crcResult >> 8 ^ crcTable[((crcResult & 255) ^ byt[i]) & 255]);
}
int crcFront= ~crcResult;

int k=bytLen - lOffset -3;

int cr1,j;
if(k>0)
{
 char *backbyt;
 backbyt=byt+lOffset+4;
 cr1=~lcrc;
 for(j=bytLen;j>bytLen - k;j--)
 {
  for(i=0;i<=255;i++)
  {
   if((cr1 & 0xff000000)==(CRCdata[i] & 0xff000000))
    break;
  }
  cr1=cr1 ^ CRCdata[i];
  cr1=cr1<<8;
  cr1=(i ^ byt[j]) | cr1;
 }
}
else
 cr1=~lcrc;

for(i=0;i<=3;i++)
```

```
 for(j=0;j<=3;j++)
 {
  for(i=0;i<=255;i++)
  {
   if((cr1 & 0xff000000)==(CRCdata[i] & 0xff000000))
     break;
  }
  cr1=cr1 ^ CRCdata[i];
  cr1=cr1<<8;
  cr1=i | cr1;
 }
 retlCRC= cr1 ^ crcResult;

 return crcFront;
}
```