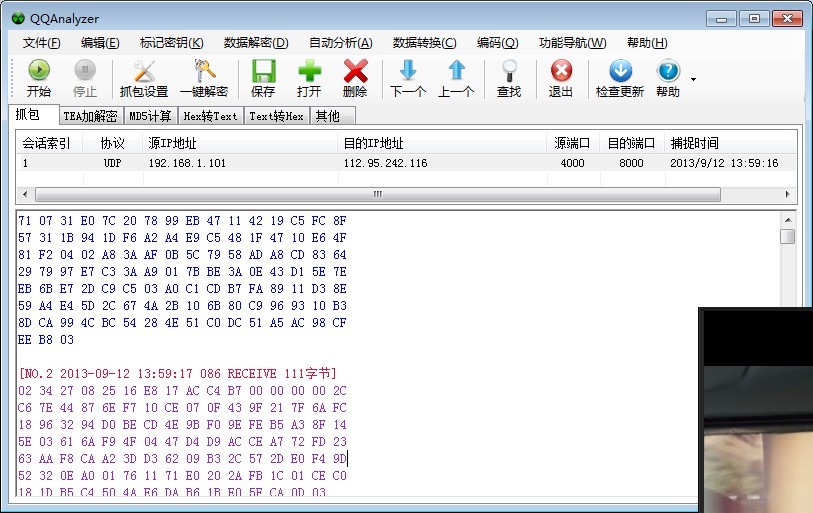
## 概述

    很多朋友都对PCQQ很感兴趣，大概都源于对这只企鹅的喜爱吧。PCQQ协议之所以这么多人想去研究，除了研究本身能带来的满足感之外，最主要还在于这种协议的挑战性。  
    那么，PCQQ的分析要从那里开始呢？这是新手们都会问的一句话。别急，我也是新手。不过这方面我还算有稍许的经验。不过这个PCQQ嘛，本身没有什么技术含量，所以，不要有什么心理包袱。再所以，请大家不要直接问我有没有PCQQ的源码，这样很不礼貌，对吧。呵呵！  
    好了，说起PCQQ，我们就不得不提到套接字(socket),TCP/IP,UDP，这些概念说难不难，说易也不易，要看我们的理解程度了。具体这些概念大家还是看一下百度吧，那儿有专家们的权威解释。套接字是两台主机之间的通信约定，具体是协议规则不是某一种编程语言所特有的。也就是说在windows平台下，我们要使用套接字只要调用平台帮我们封装好的win32 API就行了.而在各自的编程语言里，官方都会在运行的支持库或框架里封装好适合自己语言风格的类库，供代码里调用。  
    在这里，我以c#为例，给大家说一下怎么开始我们的PCQQ之旅吧！  
    第一步：  
    打造适合自己的UDP封装类。  
   UDP类封装遵循以下流程：绑定端口(这一步可以省去)，连接服务器，完了之后接收消息。  
这里我给大家参考一下我一个项目里的代码，但是大家要自己理解后写成自己的代码，否则咱们就没有下文了。

PCQQ研究第二步：  
   抓包工具，是指的能过滤成udp协议并且指定为tx服务器目标端口为8000的协议嗅探工具。比较出名的有wireshark，还有专门研究QQ类协议的 QQAnalyzer.这写软件都能很好的抓取并辅助分析出协议来。  
    两款软件在网上都有破解版可以下载到，因此，我们这里就直接用上 QQAnalyzer 3.1破解版。（360报毒，最好在研究的时候从压缩文件解压缩出来，用完之后用360删除，毒不可不防，管他有没有毒）  
    另外就是PCQQ的版本选择了。目前QQ的版本琳琅满目，但是最近据说QQ2011以及以下的版本都正式停掉了，因此，我们这能选网上分析资料相对较少的QQ2012或者2013了。我们就选QQ2013正式版SP1吧，反正2012的版本和2013的相差不太大。选最新的也没什么好顾虑的。  
    最后，为了避免其他软件产生的协议影响，比如QQ管家或是QQ音乐，反正最好是运行的软件越少越好。  
  
首先，运行QQAnalyzer软件之后，  
http://pic2.qnpic.com/doimg/UU102/b42ed40f/  
点抓包设置，按如图进行选择  
  
  
选择好之后便可开始抓包。

我们在开启抓包软件之后，要做的第一件事，就是打开QQ，开始一遍完整的QQ登录。  
   等登陆上去之后立刻将QQanalyzer停止。  
    以下图片是我抓到自己QQ之后截取的。



从上面有清晰看出我们要分析的地方。  
源IP地址：就是本机在所在的局域网对应的对外ip  
目标IP地址：腾讯的服务器ip。这个ip并不是一成不变的，每次登录都会为当前登录的QQ帐号寻找最适合的服务器。  
源端口：是本机对外开放给服务器的端口，不一定是4000，但是一般QQ的客户端是从4000开始的，如果被其他软件占用了端口，会在4000基础上加1，依此类推。直到累加后的端口没有被其他软件占用。 我们在程序里的端口之所以绑不绑定这些端口都无关紧要就是这个原因。  
目的端口：腾讯的服务器的开放端口，固定为8000.当然手机协议或其他协议则另当别论。一般我们在抓包软件里过滤协议就是以这个端口为特征而过滤的。  
  
接下来，就是我们要研究的重点了。  
  
本机与服务器通讯所产生的每一个包体都是以十六进制方式传输的，我们在抓包软件看到的仅仅是将十六进制数据的副本以文本形式显示出来而已，方便我们查看和分析。

NO.X 这里的X是指的过程中产生的包的序号标记，这是qqAnalyzer自己标记的，我们可以不用管他。  
SEND 147字节，是指的向服务器发送了147个字节。说明这个包是我们发送给服务器的包。  
  
再接下来，就是我们的重头戏协议规则分析了。先在这卖个关子，待留在下文继续披露。

## 0825消息包 发送

 一文里，我们了解PCQQ的大致信息。这里我们开始将具体包体的分析。  
    由于QQ2012之后的版本变化太大，在2012这一个版本出来之前，接触包的命令是0091，不过这都成为了过去式了。现在的接触包是0825，也就是说2012之前协议的研究都没用了。那么我们说的这个0825也好，0091也好是指的什么呢？下面我们来看之前我看到的一个包体：

[NO.1 2013-09-13 15:23:32 947 SEND 147字节]  
02 34 27 08 25 31 60 17 AC C4 B7 03 00 00 00 01  
01 01 00 00 66 7E 00 00 00 00 17 AD 8D B4 DE B3  
A9 18 C9 EA 59 46 C9 91 B5 81 AD 91 65 55 44 2C  
B6 40 49 17 F1 B0 F3 6E F6 65 44 04 18 63 0C 0F  
8A A1 DD 2A F8 B0 4E 3C C1 FD F8 E6 B6 2A B7 86  
10 02 B5 BA 62 9B 9A E9 1E 65 09 73 29 4E EE 80  
A3 90 20 5D 10 1F 85 F2 A4 5D 19 53 58 11 E5 8A  
7E 13 59 0C F1 34 60 E6 51 EE 7B F7 CE C6 B8 32  
C4 97 55 61 FD 03 D3 B3 E3 25 EB 4A F0 55 B1 11  
2C 0F 03

这里我们从 02...这里开始，算是我们的正式包体。  
首先讲一点，QQ协议的所有包体都有一个很明显又固定的特征，那就是它每一个包体的未加密部分基本相同，加密部分也能通过未加密部分找出密钥来解密包体。从我们多次抓包对比后分析可以大致得出，发送包所有的包体头部特征基本一致，接收包也是一致，只是与发送包的不一样而已。

### 消息包格式:

包头(一个字节,0x02)+版本号(两个字节)+消息号(两个字节)+消息序号(两个字节)+QQ号码(四个字节)+固定字节(N个,不同的消息数量不同)+加密数据(N个)+包尾(一个字节,0x03)

02  是用于区分包体的开始位置。相应整个包还会以03作为包体结束的标志。（固定）  
34 27 代表QQ客户端采用的协议版本，不同的版本即便是一次小小的升级都会让这个标志变化。（固定，每个版本会不同）  
08 25 重点来了，这就是我们刚才0825来0825去的所说的地方。也就是一个包的功能命令。代表这个包用于完成客户端与服务器的一次握手接触。（固定）  
31 60 包序号，用于区分每一个是独特特征。（一般固定，发送消息的包除外）  
17 AC C4 B7 这个是我的QQ号码的十六进制数，具体怎么算的稍后我会带上。（按QQ号来）  
03 00 00 00 01 01 01 00 00 66 7E 00 00 00 00 这个字符串具体什么意思不太清楚，但是每个发送包都含有，姑且我们称他为 0825\_fix0 吧（随便大家怎么叫，知道是这个数据就行了） （这里是固定的，差不多每个包都会用）

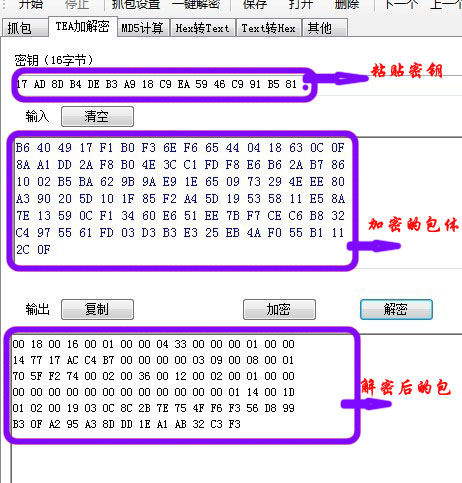
### 固定字节:

03 00 00 00 01 01 01 00 00 66 7E 00 00 00 00

17 AD 8D B4 DE B3  
A9 18 C9 EA 59 46 C9 91 B5 81 AD 91 65 55 44 2C  
B6 40 49 17 F1 B0 F3 6E F6 65 44 04 18 63 0C 0F  
8A A1 DD 2A F8 B0 4E 3C C1 FD F8 E6 B6 2A B7 86  
10 02 B5 BA 62 9B 9A E9 1E 65 09 73 29 4E EE 80  
A3 90 20 5D 10 1F 85 F2 A4 5D 19 53 58 11 E5 8A  
7E 13 59 0C F1 34 60 E6 51 EE 7B F7 CE C6 B8 32  
C4 97 55 61 FD 03 D3 B3 E3 25 EB 4A F0 55 B1 11  
2C 0F   
这一长串数据是什么意思呢？没错，肯定是加密后的包体，但是既然是要加密，起码也要看到密钥吧。至于用什么加密算法加的密，IT前辈已经帮我们OD逆向好了，知道这是采用64轮的tea算法。而且算法都已经为我们准备好了。  
那么密钥在哪呢？tea算法采用的密钥长度是16位，也就是说，在这一堆数据里，肯定有一个密钥以明文方式存着，而且不可能存在包体后面，因此加密之后的包体长度是不确定的，包体有多长谁也不敢确定，所以最可能就是放在了包体的前面。事实证明，前面的16位字节恰好是能解开后面数据的密钥。也就是这个包的密钥是：17 AD 8D B4 DE B3 A9 18 C9 EA 59 46 C9 91 B5 81   
带解密的包体是

AD 91 65 55 44 2C  
B6 40 49 17 F1 B0 F3 6E F6 65 44 04 18 63 0C 0F  
8A A1 DD 2A F8 B0 4E 3C C1 FD F8 E6 B6 2A B7 86  
10 02 B5 BA 62 9B 9A E9 1E 65 09 73 29 4E EE 80  
A3 90 20 5D 10 1F 85 F2 A4 5D 19 53 58 11 E5 8A  
7E 13 59 0C F1 34 60 E6 51 EE 7B F7 CE C6 B8 32  
C4 97 55 61 FD 03 D3 B3 E3 25 EB 4A F0 55 B1 11  
2C 0F

我们放到 QQAnalyzer的QQtea加密选项卡里



就是这样，将解密出来的包的文本再复制到我们刚才的包文本下面。  
解密出来的数据如下：  
[  
00 18 00 16 00 01 00 00 04 33 00 00 00 01 00 00  
14 77 17 AC C4 B7 00 00 00 00 03 09 00 08 00 01  
70 5F F2 74 00 02 00 36 00 12 00 02 00 01 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 01 14 00 1D  
01 02 00 19 03 0C 8C 2B 7E 75 4F F6 F3 56 D8 99  
B3 0F A2 95 A3 8D DD 1E A1 AB 32 C3 F3  
]  
那这些又有什么规律呢？还是老办法，多抓几次包，重复操作，对比一下便知：

### 解密内容

[  
00 18 00 16 00 01 // 固定,我们取名为 0825\_fix1 ，因为后面我们还有好几个地方要用到  
00 00 04 33 00 00 00 01 00 00 14 77 //固定，同上，取名为 0825\_fix2  
17 AC C4 B7 //QQ号码  
00 00 //第几次发送0825这个包，就显示第几次，第一次为0，以此类推。为什么是第几次呢，因为有可能要提示要更换服务器登录  
00 00 03 09 00 08 00 01 //固定，具体含义不清楚  
70 5F F2 74 //这里是IP地址是16进制数，[ IP地址：112.95.242.116]为什么是这样，经验吧！包抓多了就知道了  
00 02 00 36 00 12 00 02 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 14 00 1D 01 02 //固定，具体含义不清楚  
00 19 //这个代表接下来会有0x19（换成10进制是25）个字节要出现，具体作用看后边  
03 0C 8C 2B 7E 75 4F F6 F3 56 D8 99 B3 0F A2 95 A3 8D DD 1E A1 AB 32 C3 F3 //25个字节，这里是椭圆曲线加密算法的一部分。HDKey.怎么知道，OD逆向分析知道有这么一个函数在这个位置，然后查资料得到的 （为了能解决以后的验证码问题，这里即使能固定也别固定，可以从下面的链接<http://www.10qf.com/tools/pcqq.aspx>

以上分析到了0825的发送包。  
出于篇幅和时间的考虑，0825的接收包将另起一页来分析。希望大家多多支持本站。

## [0825消息包 接收](http://www.10qf.com/forum.php?mod=viewthread&tid=28&extra=page%3D1)

  我们在   [**PCQQ UDP协议的研究<2>——0825 发送包**](http://www.10qf.com/thread-27-1-1.html)  一文里了解了0825发送包的结构。那么接收包又是怎么样的呢？  
先来看我们的接收包：

[NO.2 2013-09-13 15:23:32 993 RECEIVE 111字节]  
02 34 27 08 25 31 60 17 AC C4 B7 00 00 00

AE CE  
37 33 D8 02 6E 30 4F 2B C7 88 15 A1 ED B8 FD FC  
07 80 97 45 69 D3 88 AF E1 63 88 73 86 B3 89 3D  
33 B5 B7 94 16 9F 51 0D 2E BD E4 3D 2B 00 5D 3B  
FE C2 23 58 2A 17 69 F7 6F 23 5F D6 64 F4 56 BD  
56 B3 D8 B8 BE 74 37 4A 41 B6 CC FA BD 91 E1 07  
18 1A 52 79 33 5F 54 D3 04 41 9B D3 58 BC 03

这个包体看起来是不是与刚才我们的发送包差别很大。对了，不管他差别大多少，总之他们都是有规可循的。只要找出规律，一切都是浮云。还 重复枯燥的抓包。然后分析。

02 //包体头部标记。  
34 27 //QQ协议的版本  
08 25 //接收包的命令大部分都跟发送包一一对应。没对应那就是怪胎。  
31 60 //包序号，跟发送包一致，只要能区分不同的接收包就行  
17 AC C4 B7 //QQ号码，我的。397198519

00 00 00 //3个0.这里固定

### 固定字节: 00 00 00

这里我估计也没什么难度。大家一看便知！  
接下来还有一大块呢。是什么？

AE CE 37 33 D8 02 6E 30 4F 2B C7 88 15 A1 ED B8  
FD FC 07 80 97 45 69 D3 88 AF E1 63 88 73 86 B3  
89 3D 33 B5 B7 94 16 9F 51 0D 2E BD E4 3D 2B 00  
5D 3B FE C2 23 58 2A 17 69 F7 6F 23 5F D6 64 F4  
56 BD 56 B3 D8 B8 BE 74 37 4A 41 B6 CC FA BD 91  
E1 07 18 1A 52 79 33 5F 54 D3 04 41 9B D3 58 BC

当然是加密后的数据。这次比较纯了，没有一点杂志，也没有我们想要的解密的密钥。不信你复制前16个字节重复发送包解密那一段的步骤试一下。那密钥在哪呢？对了，我们何不用之前我们加密0825发送包的那个密钥试一下？说不定还能碰运气成功呢？  
看样子我们运气不错，刚好被那个密钥给解开了。  
二话不说，先把他解析完再说吧！

[  
00 01 12 00 38 77 D6 9D 0D 3B E2 84 E4 BC 07 7B  
AF 3B 48 08 E2 3D 1B AB 28 F3 F2 7F 18 88 28 B8  
A0 6E EB 20 27 B3 F3 C5 A8 2C E3 BD 3A 44 E3 0C  
A1 F3 7C DB 64 0C 82 9A 84 28 A9 D2 69 00 17 00  
0E 00 01 52 32 BD 84 2A 30 40 5F F2 EE 00 00  
]

下面我们来解析:

### 解密内容:

[  
00 //这里意思是无需更换登录的服务器。也就是无需重定向。要重定向的话，这里就是01了。  
01 12 //固定。  
00 38 //令牌长度。标识着接下来的字节数为56(0x38）个字节。

77 D6 9D 0D 3B E2 84 E4 BC 07 7B AF 3B 48 08 E2  
3D 1B AB 28 F3 F2 7F 18 88 28 B8 A0 6E EB 20 27  
B3 F3 C5 A8 2C E3 BD 3A 44 E3 0C A1 F3 7C DB 64  
0C 82 9A 84 28 A9 D2 69 //这一行以上到00 38那个位置的数据我们取名为 0826\_token。令牌负责一个小功能，但是我们只能当成是黑箱操作

00 17 00 0E 00 01 //固定.具体含义未知  
52 32 BD 84 //这个得凭经验。多了自然就知道了。[ 时间：2013/9/13 15:23:48]  
2A 30 40 5F [ IP地址：42.48.64.95] //当前登录的服务器ip  
F2 EE //本机局域网对外端口：62190 //每台机器产生的结果不一定相同  
00 00  
]

到这里为止，是0825的接收包的解析。至于这些数据怎么去传输。我将花一篇专门的文章来说明。大家目前阶段最主要的任务就是将QQ协议分析好。怎么传输这些都是小儿科的问题。好了，欢迎每天都关注本站，多多支持本站!

## [0826 发送包](http://www.10qf.com/forum.php?mod=viewthread&tid=40&extra=page%3D1)

   距离上一篇文章 [PCQQ UDP协议的研究<3>——0825 接收包](http://www.10qf.com/thread-28-1-1.html) 已是两天之久，各位可能迫不及待想了解0826的发送包结构如何。别急，今天我肯定是有备而来。  
    首先我们将前天抓的包放上来：

02 34 27 08 26 7D E0 17 AC C4 B7 03 00 00 00 01  
01 01 00 00 66 7E 00 00 00 00 95 FA 80 BD CD B4  
69 FD 52 91 CD 21 3A 1E 95 BA 51 7F 28 C0 16 AE  
86 97 E0 5F C7 ED 6A 9B B2 E4 0B 0E C0 42 17 86  
2D AD FE 65 8E 3E 8D C9 46 8D F7 59 41 3B 5A 86  
19 61 1D 17 82 6B EB ED 25 2E D2 0B 0E DA 97 9A  
CE DE 9D 8F 25 EE BC A2 CF B3 43 E8 C4 67 34 EC  
C5 C0 3F 08 09 9B B7 AB 83 D1 14 E4 CF 5A FE 8B  
83 19 DB BB DF 8A 2F AE 97 B2 C2 F9 2C 29 42 D7  
AB DA 12 5C 9E C5 1E AB 6B 9C BE 21 26 FB AB 91  
26 0E A5 F2 0A 48 54 28 3E A5 8F A8 D4 C9 99 8C  
B1 F0 BA 70 04 02 12 00 21 AA B7 F6 D6 74 BD EB  
8F A2 E0 95 47 E2 42 73 F6 0A DD 1F 10 80 74 31  
F3 65 D2 2F B4 64 F7 A5 40 FA 1C 0C 58 08 BF 56  
CD 46 0C 02 D0 6D 07 4D F7 F2 E0 AB 45 AB 0A 37  
BE 84 82 99 7D 7D 56 AD 4C ED B6 B6 25 EA 2B F8  
FC D7 85 F0 7C CC FF 99 DD F2 DB 76 CB 13 49 C6  
6D 1B FD 9A 99 E9 5C D4 F4 A2 DF DE 81 87 B8 B1  
F4 BD 9A 27 96 F7 06 62 78 81 70 4B 66 91 A8 A6  
B0 3D 06 F4 90 4E 78 87 23 96 89 65 F1 10 B1 80  
A6 EF 5F 18 24 E7 DA 14 B6 01 B3 AF A3 4B FD AE  
D4 F3 15 05 00 42 02 16 A7 8E 32 62 DB 85 F9 8E  
17 2A 20 9C 9B 13 D1 47 BC A8 75 96 B3 D9 A4 91  
16 E7 F1 4A 6D D7 C2 F3 48 D2 27 D3 58 5E 55 57  
DA 29 DA AD 76 7F 12 91 54 C0 18 7B DC E9 26 D4  
06 46 D6 81 11 56 7C B1 51 93 B8 32 9C 37 2E EC  
91 13 95 34 40 7F 19 4E 35 13 83 CA 50 26 C1 82  
15 1C 27 B1 77 9A BD 92 45 87 9E C5 8E 95 C4 7B  
27 BC CE 2F 24 2A F3 16 70 F7 58 15 9D F5 5F 5D  
E5 2E F9 80 B4 FC 8C 4B 14 EC 26 0A 23 C2 AA 85  
AF 7D 5C 99 58 F8 8B 94 07 48 84 75 28 FD BD 4F  
FD D5 C2 80 F2 6C 48 24 62 5C 93 53 D7 EC BA 8B  
40 46 54 E7 AD 59 AA FF 90 AC 2A 37 E3 1C D6 F5  
94 26 54 87 19 CD 3F AC 24 47 C7 99 8C D7 F5 13  
18 96 14 39 B9 69 A4 D8 06 12 6A 66 D0 2E F7 3E  
7B C6 B0 45 2B A4 32 5D 8A F7 82 3B 50 E0 F3 FD  
6D AD E3 DC C1 3F 4E 26 6D 4C 7D 72 62 20 4F 70  
A5 EA FD 73 27 DA 0D 28 EA 87 03

    还是老样子，包体的头部如下：

02 34 27 08 26 7D E0 17 AC C4 B7 03 00 00 00 01  
01 01 00 00 66 7E 00 00 00 00

这个没什么悬念，大家都知道分析了，接下来的密钥还是跟找0825那个包一样的找，很轻松就发现接下来的16个字节就是0826的密钥。

95 FA 80 BD CD B4 69 FD 52 91 CD 21 3A 1E 95 BA

下面是解密出来的结果：

[  
01 12 00 38 77 D6 9D 0D 3B E2 84 E4 BC 07 7B AF  
3B 48 08 E2 3D 1B AB 28 F3 F2 7F 18 88 28 B8 A0  
6E EB 20 27 B3 F3 C5 A8 2C E3 BD 3A 44 E3 0C A1  
F3 7C DB 64 0C 82 9A 84 28 A9 D2 69 00 05 00 06  
00 02 17 AC C4 B7 00 06 00 78 10 C1 88 6C BA 71  
47 0E A8 44 69 DE 50 8F 5D E6 8F 44 8D 8F 14 70  
30 F0 9B 80 0F 1D C0 5C B6 F2 22 35 D2 7C 9C 24  
56 59 AA A6 E8 14 E8 67 68 9C 2A 31 0B 5C 81 6F  
02 0F B3 D1 E1 F7 2F F6 7E 35 D6 C1 BC 17 C8 18  
5B D9 B5 8A 95 F8 90 77 1C 3A AF 48 F1 FC 73 BC  
D9 99 32 F3 5C 49 81 ED E6 5D 2F 7C 86 8E C7 29  
2A 39 4B BC 49 AF AF 14 AF 99 E0 A6 83 90 68 50  
9E 34 00 1A 00 40 BE 40 66 A6 6F 29 89 59 81 52  
DB 99 BA 65 10 59 84 24 A5 63 92 1B 32 52 1C 87  
F3 D8 31 E0 29 12 A7 07 54 78 B3 D4 8C C1 F4 B9  
4A 35 E3 1E 0E 78 A6 62 94 65 4D 2A 91 88 AF CC  
4F 5D 1C 9F 5F 3B 00 18 00 16 00 01 00 00 04 33  
00 00 00 01 00 00 14 77 17 AC C4 B7 00 00 00 00  
01 03 00 14 00 01 00 10 27 35 20 5A AA 76 CB D0  
EE AF F4 BA 14 4A 71 0F 00 32 00 63 3E 00 63 02  
03 05 01 00 00 04 00 7B D4 00 00 00 00 AD D9 9C  
D2 8B 81 57 4C 35 EC 74 A8 B7 57 0C 9A 03 01 00  
00 66 7E 14 77 00 00 00 00 00 00 00 00 07 DD 00  
08 00 14 00 01 00 02 00 03 00 00 00 00 00 89 9D  
DC 06 B2 92 17 3A 69 DA 22 4D 7F 6C 4F 2D 0F 41  
43 E7 8C 6D 2C CC 5B 41 DF 9C D7 F4 6D C9 68 01  
14 00 1D 01 02 00 19 03 0C 8C 2B 7E 75 4F F6 F3  
56 D8 99 B3 0F A2 95 A3 8D DD 1E A1 AB 32 C3 F3  
01 02 00 62 00 01 2F F4 BE 05 6E E7 10 CA 2C C6  
64 3C D0 2F 6E 8A 00 38 DA 6C E3 EA BF F5 E9 39  
F3 35 CC E1 47 91 F2 00 75 A7 11 C7 F9 D3 9F 20  
23 F8 8A 93 6B 7D 2A 6A 04 95 74 13 B4 C4 70 C7  
E2 9F ED 69 F1 D3 73 B1 CC 68 79 89 C2 47 D8 B7  
00 14 52 FA 2B 88 BF 30 D2 BE 3A 91 B8 BF B5 02  
33 7F 55 D2 B4 25  
]

这个包数据量真不少啊，tx的砖家们可真舍得花时间想出这样的构思啊。这是真正的为中国软件行业（尤其是软件安全领域）作出巨大的贡献。  
下面我们还是咬紧牙关来分析一下：

### 解密内容:

[  
01 12 //固定  
00 38 //这个是之前0825接收包里的东东。 0826\_token

77 D6 9D 0D 3B E2 84 E4 BC 07 7B AF 3B 48 08 E2  
3D 1B AB 28 F3 F2 7F 18 88 28 B8 A0 6E EB 20 27  
B3 F3 C5 A8 2C E3 BD 3A 44 E3 0C A1 F3 7C DB 64  
0C 82 9A 84 28 A9 D2 69   
00 05 00 06 00 02 //固定

17 AC C4 B7       //[QQ号码：397198519]  
00 06 //固定

00 78 //转化为10进制是120个字节，这里面包含md5密码等数据,

//密钥是: md5(md5(pwd) + 00 00 00 00 + QQ)  
10 C1 88 6C BA 71 47 0E A8 44 69 DE 50 8F 5D E6  
8F 44 8D 8F 14 70 30 F0 9B 80 0F 1D C0 5C B6 F2  
22 35 D2 7C 9C 24 56 59 AA A6 E8 14 E8 67 68 9C  
2A 31 0B 5C 81 6F 02 0F B3 D1 E1 F7 2F F6 7E 35  
D6 C1 BC 17 C8 18 5B D9 B5 8A 95 F8 90 77 1C 3A  
AF 48 F1 FC 73 BC D9 99 32 F3 5C 49 81 ED E6 5D  
2F 7C 86 8E C7 29 2A 39 4B BC 49 AF AF 14 AF 99  
E0 A6 83 90 68 50 9E 34

[  
28 25 5A 45   
00 01   
17 AC C4 B7  //[QQ号码：397198519]   
00 00 04 16 00 00 00 01 00 00 13 9F   
//这里不是0825\_fix2 ,00 00 04 33 00 00 00 01 00 00 14 77，固定就行，每个版本都不同，但是这里似乎有点规律，留待各位自己慢慢找吧！  
00 00 01 //固定  
16 C7 ED 01 20 3C 89 EC 48 A1 CE CB A3 7C 41 FB//用密码做一次md5加密得到的结果  
52 2B C7 00 [ 时间：2013/9/8 8:38:24] //QQ号码在当前服务器最后一次登录的时间。  
//（注意每次登录的服务器有可能不一样,所以即使刚才我们才登录过一次，上一次登录也不一定就是今天啦）   
01   
52 09 A6 D3[ 时间：2013/8/13 11:24:03] //当前登录的时间   
00 00 00 03 00 00 00 00 //固定，但是不是绝对的。这个03的位置可能是别的数  
2A 30 34 C9 [ IP地址：42.48.52.201]  
00 00 00 00 00 00 00 00 00 10 //固定  
F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 F0 //固定  
40 7D 35 59 36 59 5D 66 2B 36 2A 25 2A 4A 43 53 //密钥，取名为0826\_key1,区分之前我们用的密钥  
//这个密钥会用在那个00 40令牌的解密，还可能会有其他用途  
//果不其然，这个密钥还能解开0826的接收包，哈哈，神奇吧！  
]   
00 1A //固定  
00 40 //加密后的CRC数据。关于CRC的算法，[www.10qf.com](http://www.10qf.com/" \t "_blank)论坛上有具体文章  
BE 40 66 A6 6F 29 89 59 81 52 DB 99 BA 65 10 59  
84 24 A5 63 92 1B 32 52 1C 87 F3 D8 31 E0 29 12  
A7 07 54 78 B3 D4 8C C1 F4 B9 4A 35 E3 1E 0E 78  
A6 62 94 65 4D 2A 91 88 AF CC 4F 5D 1C 9F 5F 3B  
[  
00 15 00 30 00 00 //固定，00 30预示后面有0x30个字节的数据  
01 //固定，代表第一个CRC数据的序号  
3E 29 AA EE //CRC Code取反,意思就是说将字节的排列顺序完全反过来，下同,我们取名为 crc\_code0  
00 10 //代表后面有16个字节，为crc待验证字节。取名 crc\_data0  
EA F8 F4 9C 00 8B 3A C2 55 F7 78 D5 E7 C5 20 63  
02 //固定，crc序号  
56 73 44 C4 //crc\_code1  
00 10 //同上解释  
AD D9 9C D2 8B 81 57 4C 35 EC 74 A8 B7 57 0C 9A crc\_data1  
]  
00 18 00 16 00 01  //0825\_fix1 ，之前在0825发送包里见过的，就不记得了？呵呵  
00 00 04 33 00 00 00 01 00 00 14 77 //0825\_fix2  
17 AC C4 B7   //[QQ号码：397198519]   
00 00 //不一定是00 00 ，有可能是00 01  
00 00 //一般固定  
01 03 00 14  00 01   
00 10 //16字节的令牌，具体怎么来的，暂时硬编码吧，反正对比了好几次都是固定的  
27 35 20 5A AA 76 CB D0 EE AF F4 BA 14 4A 71 0F//固定  
00 32 00 63 3E 00 63 02 03 05 01 00 00 04 00 //固定，版本不同而不同  
7B D4 //每次都变化，当他是随机两字节  
00 00 00 00 //固定，但是也不是绝对的  
AD D9 9C D2 8B 81 57 4C 35 EC 74 A8 B7 57 0C 9A  
03 01 00 00 66 7E 14 77 00 00 00 00 00 00 00 00  
07 DD 00 08 00 14 00 01 00 02 00 03 00 00 00 00   
00 //这里开始往上49个字节,都是固定的  
89 9D DC 06 B2 92 17 3A 69 DA 22 4D 7F 6C 4F 2D  
0F 41 43 E7 8C 6D 2C CC 5B 41 DF 9C D7 F4 6D C9  
//可悲的32个字节，没找出规律来，据说是32个字节的机器码，这里暂时硬编码  
68 01 14 00 1D 01 02 //固定  
00 19 //0825\_token 这个00 19是不是很眼熟，对了，就是我们刚刚在0825发送包里的那一段数据，用于ECC密钥传输  
03 0C 8C 2B 7E 75 4F F6 F3 56 D8 99 B3 0F A2 95  
A3 8D DD 1E A1 AB 32 C3 F3  
01 02 00 62 00 01 //固定  
2F F4 BE 05 6E E7 10 CA 2C C6 64 3C D0 2F 6E 8A//这个16个字节具体未知  
00 38 //未知功能的一个令牌，当他是随机好了，数据我们也随机生成，  
//要不保险一点就直接硬编码（把这下面的数据直接固定到我们程序里）  
DA 6C E3 EA BF F5 E9 39 F3 35 CC E1 47 91 F2 00  
75 A7 11 C7 F9 D3 9F 20 23 F8 8A 93 6B 7D 2A 6A  
04 95 74 13 B4 C4 70 C7 E2 9F ED 69 F1 D3 73 B1  
CC 68 79 89 C2 47 D8 B7   
00 14 //CRC校验的数据  
52 FA 2B 88 BF 30 D2 BE 3A 91 B8 BF B5 02 33 7F //CRC数据1  
55 D2 B4 25 //CRC code值  
]

   到这里为止，我倒吸了一口凉气，要是tx哪天再在后面加上一两个这样的包体，我们的半条命就终结了！好了，0826的发送包讲完了，里面可能还有一些未解答的秘密，留待大家一起研究！欢迎跟帖留言！  
在这里，给大家推荐一下我写的QQ解析辅助工具，可以解决一切遇到的函数功能问题：  
[PCQQ协议解析辅助工具【本站原创】](http://www.10qf.com/thread-11-1-1.html)

## [0826 接收包](http://www.10qf.com/forum.php?mod=viewthread&tid=42&extra=page%3D1)

   上一篇，  [PCQQ UDP协议的研究<3>——0826 发送包](http://www.10qf.com/thread-40-1-1.html) 解析了QQ的0826发送包。  
这一篇文章我们来分析0826的接收包。  
0826包分多种情况,比如密码错误,需要验证码,或是别打什么号码限制登录之类的..通常都无法用正常的密钥解密.判断是否是正常密钥,有如下标准:  
返回包的长度=127 : 账号异常   
                  =175是密码错误   
                  =740或799(不一定是这个数) 是账号密码均正确   
                 =871就是需要验证码  
[NO.4 2013-09-13 15:23:33 075 RECEIVE 799字节]  
02 34 27 08 26 7D E0 17 AC C4 B7 00 00 00 95 D6  
98 59 37 CD 09 44 5A E4 F6 2C 8E EF 7D 7A 95 6C  
FF 20 36 E8 5A 25 03 27 6E 61 58 DE DE 8D 65 CD  
D2 C4 83 08 FA 74 9D 70 E3 95 13 9D 60 46 A1 2A  
AA 24 EA 4E 9C 1C EC 4D 50 79 44 47 07 54 BD 6F  
86 54 5D 1A DA 71 7A 62 7F 35 57 91 5F CE E3 EE  
E5 F2 32 C5 80 7F D5 92 FC 2A DB C9 97 B5 F3 5D  
1D 09 55 FF BC 72 A6 A2 57 B2 70 B3 CB A8 0F DE  
A6 BE DA 20 65 CA 9A 77 05 EA B5 1A 29 36 EE 0D  
83 A7 26 51 F5 74 ED 71 29 B0 FB 14 F4 52 01 74  
BF 2B 73 A2 99 0E D1 33 F3 5A 42 0B 8D FC B4 E0  
E8 D3 61 8C 43 25 DC 06 19 33 DE 50 ED 21 29 E8  
D8 14 85 4B 9C D6 7B 8C 59 35 37 2A EB 1B 8D B6  
66 04 3F BE D6 8A 0B CD 37 9B 5A CC 06 FD 7B 9C  
E1 88 80 54 E2 96 39 B9 8C B1 B5 7B 35 C4 1F 42  
F9 B9 C7 EA BD 21 E6 66 1D 98 BE 09 AF D2 FE FD  
DE EC 02 53 D9 55 6F A0 46 78 8E 03 75 82 11 CE  
03 34 B9 6D 81 6F 1E 50 18 EA 29 D9 FA 77 2A A9  
36 A0 88 36 67 2E 5B 82 7F 3C 41 DF E5 3E 77 D9  
A9 7C 8A 44 E1 93 6C 5D E0 1C 64 E3 44 EB F2 64  
25 8F 9C 02 C4 D7 AC 9F CA 64 7B F3 D7 A6 22 C0  
9E DB E3 8E CC E6 66 7D BB 67 B5 28 E8 6B C8 F9  
97 07 60 F0 AA 91 96 18 88 A5 5C 64 96 8F A3 52  
E6 16 5A 02 B1 1E 20 1C 6A 67 4C 38 0B C6 07 0B  
B8 FA F2 CB 05 00 4C C7 B7 3C 48 AF E9 46 DF 77  
FD DF 45 1E FA 06 E5 69 5A 1C B2 02 F7 7E 84 80  
F2 8B 01 34 F9 ED E9 9D 31 1E 68 64 BE DE 4E 7D  
55 2A 38 87 F6 25 4C E6 63 03 4E 1E DE 67 90 18  
5B 54 AC D9 EC 9A DE 61 47 F1 0D D8 1B 81 C1 53  
D7 3B FF 16 22 8B 9A A5 C1 EC 36 1B 0E 08 FC 4D  
5F 08 5B EB EA CE ED BA A2 BD 7F CD 43 3B F4 A6  
BA 5F 9A E4 00 A1 BE 40 C8 C1 A6 F1 4D FF DC C2  
25 20 7E E7 3D 60 64 F1 11 3A C6 03 63 9D 0C 4F  
71 5E BB EF B1 30 09 37 75 C3 CE 9F A2 F7 AD D8  
9D 01 E4 5E 18 42 B8 D0 1C 5B 57 10 6B 50 01 8A  
A0 A5 6C 92 86 A3 7C 5F 80 D8 9E DB 1A 14 D3 22  
7C 79 F8 F1 1C 71 C4 67 D8 96 22 FA C2 3B 0D 2A  
EA 7D A1 B9 82 B2 04 AA EF D5 E7 2A 73 9E 8B 36  
FC 9D C8 C4 0D 1A 56 8E 8B A7 8A E0 E2 00 9E F7  
52 7D 87 98 E9 C3 98 27 79 9A 60 D7 D0 EA F1 87  
1C 68 3F 6C D4 A7 DD 44 19 7D 2D 1C 4D 9E C3 8C  
4A 04 5D 41 85 BE 64 47 06 B1 1F 77 B3 52 51 0A  
CD 7B 6C 9D 64 48 8E 28 03 CF 3F F1 7D 8F AF 23  
7F 4F 60 FF 49 0E 6D E3 9F 0C C2 3B DC 65 6F 1F  
38 D1 F6 86 93 1F 5F 33 18 F1 A2 60 6C D1 F8 67  
D7 BF 76 5B 37 0B A0 A8 9B BB 5A B3 81 C5 29 DC  
67 9A 7D 30 48 3C D8 01 BB BC FB 1D FC 66 BE A3  
BA 42 77 5E 2B 96 A3 1F 1B 42 50 E0 39 0F 45 3F  
17 C4 1A FA 0C EA 88 BD C1 02 38 11 BD 50 A9 FF  
69 87 76 C1 4D 9E 46 F0 17 91 28 83 BF 44   
  
  
    这里，我顺带提一下，之前或者说现在，分析出来的密钥也好，令牌或固定数据也好，命名都十分的混乱，我怕有一天受点刺激分不清哪个是哪个了，所以请各位别像我一样急于求成，要把我们遇到的密钥或别的有用的数据复制到专门的地方并娶一个好听一点的名字。这样我们找起来也比较方便。  
  这里的头部是：  
02 34 27 08 26 7D E0 17 AC C4 B7 00 00 00   
  
所以后面的数据（不包括03）全部是加密的数据。那么密钥是什么呢？聪明的你可能把凡是16个字节的数据全都当成密钥试了一遍，最后试出来了，但是费了很大劲。这里我还是给大家说一下，免得更多的人走弯路。  
这个解密的密钥就藏在了我们之前在0826的发送包那个0x78的令牌里。也就是那最个：  
40 7D 35 59 36 59 5D 66 2B 36 2A 25 2A 4A 43 53 //密钥，取名为0826\_key1   
因此我们把包体解开之后，变得到下面的数据：

[  
00 01 09 00 70 00 01 46 70 36 54 79 6A 70 68 63  
4B 63 67 7A 33 4B 46 00 38 EB 7E 2E 5D 72 C3 8D  
84 7F 1D BD 51 36 4E D0 A4 8F CB C1 11 2F 4F CB  
79 B3 50 8B CD E9 74 BE 73 FD FD F4 FC 95 4B 42  
5D 0C 1A A7 3E 0C 7C 80 A9 4F F6 B9 08 91 30 A5  
85 00 20 E9 43 51 59 D5 D7 C9 B6 61 DF 02 83 D1  
34 7D 58 FF 8F 0D EF 85 A5 70 39 8B 67 97 17 99  
67 87 D9 00 00 01 03 00 14 00 01 00 10 27 35 20  
5A AA 76 CB D0 EE AF F4 BA 14 4A 71 0F 01 10 00  
3C 00 01 00 38 E2 7D 42 3D 16 FE 99 AF 4D 01 A7  
10 A0 79 46 23 E2 DC 0E 32 00 F3 BD 5E F2 F3 AC  
71 47 3B 7B 60 F0 9F 6E 12 9B 6B F0 74 85 8F E2  
94 C0 A0 A4 55 DA C8 96 1F 03 6C 19 ED 00 33 00  
18 E1 17 B8 35 88 12 85 44 64 49 EC 16 89 D6 EF  
78 52 5D 44 77 B4 6D C7 96 01 07 01 D3 00 01 00  
16 00 00 00 01 00 00 00 64 00 00 0D CD 00 01 51  
80 00 00 00 3C 00 02 61 57 37 71 4A 75 76 53 54  
56 37 4B 49 50 25 7A 00 88 00 04 52 32 BD 85 2A  
30 40 5F 00 00 00 00 00 78 CB 96 C2 05 20 D4 AC  
6A DD 42 81 1A D0 A3 CB CC 74 0B 48 E3 CC CC 10  
31 0A 22 31 70 72 23 29 00 68 A8 B8 62 F7 A0 2F  
E1 B2 A5 A0 4A B4 FF 9A 67 C4 AD 62 FC C3 F9 6A  
08 37 8E 7F 6D 7D 4A 42 BA AF 91 DB D9 C9 87 69  
34 AF 2E B6 7C 1E A8 5A 44 F8 0B 14 F3 6E 8A 54  
9C 5D 97 96 E0 6B 7A 81 75 CE 40 04 BB 83 C3 3D  
F3 37 75 A4 A0 3C A3 13 AF 55 BF EA D8 43 04 75  
E3 38 5F 2D 57 64 41 60 4E 49 5E 46 44 63 2C 4A  
70 00 78 00 01 52 32 BD 85 00 70 91 BB 4B D7 50  
13 36 F6 A4 39 46 C9 1D FB 0F F5 82 A4 E9 31 C0  
1A AF 46 80 68 B4 45 CF AA AB D7 01 6F 07 30 00  
2A DF 79 FA CA F2 7E F1 23 5D 26 17 DC 7F 61 1C  
75 A6 44 3A 12 A0 13 61 12 DA D4 75 A4 1A 9A 04  
3E E7 D7 57 EE 92 4A 41 0A 43 1B 27 D1 AF 97 7E  
81 F1 12 7E B9 DE 07 89 34 62 40 FA 04 F6 BB EF  
40 0D 7C 27 7E 57 9A FC 03 A7 AE 00 83 01 59 23  
5F 7D 2B 46 74 63 23 32 63 76 37 2C 54 59 00 70  
00 01 52 32 BD 85 00 68 02 DF 5D DB DB 82 27 F6  
D4 46 21 17 46 0E F1 57 05 01 A2 F6 3A 9C 8D 8E  
A9 71 5D FE D4 60 86 37 41 71 62 21 CD FD 2B B2  
13 EA 76 27 EE 29 DF C3 C9 3A 34 EC DC 29 0C 84  
7E C0 81 2F A0 34 16 8B 9B F5 43 94 DF 36 C7 8C  
AD 9A 1B 26 7D 66 A3 0E AE 7A 7F A1 34 70 FC 4B  
E1 64 A7 28 6B 69 4B A9 9D C7 AA CE E9 37 07 E7  
7B 7D 7A 3E 7B 25 64 59 37 49 5B 2A 71 73 6D 56  
01 08 00 27 00 01 00 23 00 1B 01 1A 12 E5 87 8C  
E6 99 A8 E7 9A 84 E6 90 9C E7 B4 A2 E8 80 85 01  
13 48 82 42 19 00 04 00 00 00 01 01 15 00 10 5A  
77 9C 04 41 C9 45 0B E6 5A 8F 22 DF BE 6B 6A  
]

以下是我的分析：’

本帖隐藏的内容

[  
00 //成功  
01 09 00 70 00 01 //固定  
46 70 36 54 79 6A 70 68 63 4B 63 67 7A 33 4B 46   
//这里很容易看出是16个字节独立的，因为紧接着就有了0038这个熟客。  
因此猜测是一个密钥：经证实，它就是我们要找的0828发送包的密钥  
00 38 0826\_token1  
EB 7E 2E 5D 72 C3 8D 84 7F 1D BD 51 36 4E D0 A4  
8F CB C1 11 2F 4F CB 79 B3 50 8B CD E9 74 BE 73  
FD FD F4 FC 95 4B 42 5D 0C 1A A7 3E 0C 7C 80 A9  
4F F6 B9 08 91 30 A5 85   
00 20 //令牌，32个字节  
E9 43 51 59 D5 D7 C9 B6 61 DF 02 83 D1 34 7D 58  
FF 8F 0D EF 85 A5 70 39 8B 67 97 17 99 67 87 D9   
00 00 01 03 00 14 00 01 //此8个字节固定不变  
00 10 //只是发送包里的16个字节的令牌。有用吧  
27 35 20 5A AA 76 CB D0 EE AF F4 BA 14 4A 71 0F  
01 10 00 3C 00 01 //6个字节固定  
00 38 //这个都是在0826发送包里出现的令牌  
E2 7D 42 3D 16 FE 99 AF 4D 01 A7 10 A0 79 46 23  
E2 DC 0E 32 00 F3 BD 5E F2 F3 AC 71 47 3B 7B 60  
F0 9F 6E 12 9B 6B F0 74 85 8F E2 94 C0 A0 A4 55  
DA C8 96 1F 03 6C 19 ED  
00 33 //固定  
00 18 //24个字节的令牌，具体功能疗效不详  
E1 17 B8 35 88 12 85 44 64 49 EC 16 89 D6 EF 78  
52 5D 44 77 B4 6D C7 96  
01 07 01 D3 00 01   
00 16 //22个字节的未知作用的令牌，固定不变  
00 00 00 01 00 00 00 64 00 00 0D CD 00 01 51 80  
00 00 00 3C 00 02  
61 57 37 71 4A 75 76 53 54 56 37 4B 49 50 25 7A //16个字节的密钥，用于解开0828的接收包  
00 88 00 04 //固定   
52 32 BD 85 //当前登录的时间[ 时间：2013/9/13 15:23:49]   
2A 30 40 5F [ IP地址：42.48.64.95] //当前登录的服务器ip  
00 00 //这里可能是之前接口留下的端口信息  
00 00   
00 78 //120个字节的令牌，  
CB 96 C2 05 20 D4 AC 6A DD 42 81 1A D0 A3 CB CC  
74 0B 48 E3 CC CC 10 31 0A 22 31 70 72 23 29 00  
68 A8 B8 62 F7 A0 2F E1 B2 A5 A0 4A B4 FF 9A 67  
C4 AD 62 FC C3 F9 6A 08 37 8E 7F 6D 7D 4A 42 BA  
AF 91 DB D9 C9 87 69 34 AF 2E B6 7C 1E A8 5A 44  
F8 0B 14 F3 6E 8A 54 9C 5D 97 96 E0 6B 7A 81 75  
CE 40 04 BB 83 C3 3D F3 37 75 A4 A0 3C A3 13 AF  
55 BF EA D8 43 04 75 E3   
38 5F 2D 57 64 41 60 4E 49 5E 46 44 63 2C 4A 70 //密钥  
00 78 00 01   
52 32 BD 85[ 时间：2013/9/13 15:23:49] //解释同上，时间  
00 70 //令牌，先记着  
91 BB 4B D7 50 13 36 F6 A4 39 46 C9 1D FB 0F F5  
82 A4 E9 31 C0 1A AF 46 80 68 B4 45 CF AA AB D7  
01 6F 07 30 00 2A DF 79 FA CA F2 7E F1 23 5D 26  
17 DC 7F 61 1C 75 A6 44 3A 12 A0 13 61 12 DA D4  
75 A4 1A 9A 04 3E E7 D7 57 EE 92 4A 41 0A 43 1B  
27 D1 AF 97 7E 81 F1 12 7E B9 DE 07 89 34 62 40  
FA 04 F6 BB EF 40 0D 7C 27 7E 57 9A FC 03 A7 AE  
00 83 01   
59 23 5F 7D 2B 46 74 63 23 32 63 76 37 2C 54 59  
00 70 00 01  
52 32 BD 85[ 时间：2013/9/13 15:23:49]   
00 68 //令牌。先记着，以后可能会有用  
02 DF 5D DB DB 82 27 F6 D4 46 21 17 46 0E F1 57  
05 01 A2 F6 3A 9C 8D 8E A9 71 5D FE D4 60 86 37  
41 71 62 21 CD FD 2B B2 13 EA 76 27 EE 29 DF C3  
C9 3A 34 EC DC 29 0C 84 7E C0 81 2F A0 34 16 8B  
9B F5 43 94 DF 36 C7 8C AD 9A 1B 26 7D 66 A3 0E  
AE 7A 7F A1 34 70 FC 4B E1 64 A7 28 6B 69 4B A9  
9D C7 AA CE E9 37 07 E7   
7B 7D 7A 3E 7B 25 64 59 37 49 5B 2A 71 73 6D 56  
01 08 //固定，应该是自己的资料信息。字节长度可以确定，但是具体含义还有待挖掘  
00 27   
00 01 //这里貌似是固定的  
00 23   
00 1B   
01 1A   
12 //昵称的长度，  
E5 87 8C E6 99 A8 E7 9A 84 E6 90 9C E7 B4 A2 E8  
80 85//总共25个字节，QQ昵称：凌晨的搜索者  
01 13 48 82   
42 //好友中排名，不确定是不是这样的  
19 //年龄  
00 04 00 00 00 01 01 15 //固定  
00 10 //令牌，后面应该会用到的  
5A 77 9C 04 41 C9 45 0B E6 5A 8F 22 DF BE 6B 6A  
]

之所以中间这么会有这么多没有注释的地方，原因有两个，第一就是这是一个接收包，重要性不及发送包，唯一要做的就是检索有用的信息，把他们提取出来留着备用。第二，一开始的命名很混乱，所以我干脆这些很容易看出来的数据都不注释了，免得误导大家。有疑问的地方还是留待大家跟帖吧！

## [0828 发送包](http://www.10qf.com/forum.php?mod=viewthread&tid=43&extra=page%3D1)

上一篇“[**PCQQ UDP协议的研究<5>——0826 接收包**](http://www.10qf.com/thread-42-1-1.html) ”分析了0826接收包，分析的有点匆忙，若是还对那个包有疑问，欢迎在原帖后跟帖。

   0828可是比较关键的包了，因为通过这一步，就能拿到QQ的会话密钥（sessionkey）。会话密钥能做什么用呢？当然是解开后面大部分包的一个通用钥匙，作用自然不可小觑。能走到这一步不容易啊！  
    下面我们把0828的发送包数据搬上来：

[NO.5 2013-09-13 15:23:33 207 SEND 677字节]  
02 34 27 08 28 6F 0E 17 AC C4 B7 02 00 00 00 01  
01 01 00 00 66 7E 00 30 00 3A 00 38 EB 7E 2E 5D  
72 C3 8D 84 7F 1D BD 51 36 4E D0 A4 8F CB C1 11  
2F 4F CB 79 B3 50 8B CD E9 74 BE 73 FD FD F4 FC  
95 4B 42 5D 0C 1A A7 3E 0C 7C 80 A9 4F F6 B9 08  
91 30 A5 85 98 C4 8C 98 A2 E6 CA 06 E0 86 27 FE  
66 7E F1 32 54 1A A0 5E FF FB D0 D6 66 93 7D B8  
C4 F8 A3 7F F1 37 2B 9A 57 1B B6 6C 3B 8D F0 1B  
C1 F8 6C 79 50 5C 2F F7 9F 42 E4 90 B2 D5 D9 B5  
B3 24 4C B5 F5 FC 70 C5 16 44 58 F3 3F A6 9D C4  
48 89 34 86 25 CD 79 D6 E7 45 CA 59 98 77 12 BF  
0C 88 1A F1 58 CE 5C C5 C3 88 1B BA B2 0D A5 91  
CF CA C2 A8 11 EE 3C 9E C5 C0 8A AD F4 D2 77 E1  
45 03 16 AE 80 80 D9 6F 16 F6 02 59 C9 FE 02 55  
F7 78 B6 F6 70 63 A9 35 5B CA 07 11 7C CD 2A 28  
CD 79 43 3D BF FD 7D 06 D6 7F CA A5 43 F4 A8 3C  
28 35 57 5C B9 B0 53 5E 56 E0 B3 32 88 6B C0 39  
D6 24 7F 2D 82 AF 06 0B 48 27 57 03 0A 6A 3F 98  
7B 5B EE 97 32 9D FB EB 16 AA 95 B4 7D F1 F2 11  
44 1A C3 97 EC 02 03 0A F3 37 7C EC 3F 37 37 A9  
BD 0F 90 6D 6B 66 61 B4 54 3C D1 29 02 95 4F 6A  
02 56 3E 64 43 EE D2 CD C6 29 3B 21 B9 96 FB DF  
05 C5 87 8A FC C4 5C 98 82 E6 A1 E0 93 66 9B DE  
C2 71 A8 07 4D D4 02 5C B9 74 A8 D8 F0 DC AB A4  
59 26 2E E5 FE CD EF 03 A6 3B 23 B3 30 24 A7 9B  
32 3D AD B1 85 3E EF 41 CC E5 C7 77 E3 95 F0 ED  
63 ED 10 AB E4 3F D2 10 DF 04 2B 5D 06 B1 C1 53  
AB D5 BF 91 1E 1D 25 F9 2E EF 65 C7 FC 12 54 73  
9D 90 C7 6A 20 97 DA 6A 8E 5A 58 72 D6 D7 57 0A  
49 3C D4 6F 73 9E BF 74 5C CD 15 45 39 7A 5E 11  
72 A0 C9 10 06 24 0F 02 6A CB 92 EA 85 D9 4E 0A  
F4 5D 8D F0 F6 6D 8E 6F 57 99 3F 28 19 70 48 B0  
66 F5 F5 AF 8E 28 DF F2 09 D1 65 A9 14 2E 6E 1F  
43 46 47 38 B3 30 9E D4 74 C1 28 2F 58 93 87 61  
95 3D AB DC 67 0E B9 58 E2 72 74 EE D4 C7 3D 06  
25 6B 6F 06 16 C3 6E DC AC FE 76 FB 15 4C 61 5E  
05 8D BD 6E 33 0A D4 4C 29 BD 84 75 4F 07 1D A0  
70 9D A7 33 77 AD B9 76 AB C6 9C C4 EE 3A 30 D9  
8C 42 C6 E5 1B 23 09 AA 70 9E DD 24 AD 7E CE D0  
97 88 E4 E1 FC 7D 2F AB 69 EF 80 58 CF 3C D4 1C  
69 E0 3E BD 62 9B 63 FD 36 76 00 7F 8A 5C A0 58  
F3 90 4E D4 0C 33 E0 AC B3 51 2D 6E 2E 53 47 90  
25 7A 7F F0 03

这个包除了比较关键，还比较特殊，为什么呢？大家看他的头部和前两个发送包的头部有一点不一样吧？我们来对比一下与0826的发送包头部的差别。

02 34 27 08 26 7D E0 17 AC C4 B7 03 00 00 00 01  
01 01 00 00 66 7E 00 00 00 00  //0826的发送包  
  
02 34 27 08 28 6F 0E 17 AC C4 B7 02 00 00 00 01  
01 01 00 00 66 7E 00 30 00 3A //0828发送包

仔细对比一下差别，这里是千万要注意的，我们0828的头部不一样。  
     接下来，继续看下面的数据，发现有一个00 38开头的数据。这不是一般令牌的字节长度标识吗？怎么会在这出现呢？难道后面接着的就是一个令牌？没错，细心的童鞋可能已经发现，这个00 38的令牌恰好是我们在0826的接收包里出现过的令牌。巧合？+必然！  
    再接下来，就是完全没有规律的数据了，这下该是加密了的数据了吧！我们用哪个密钥来解密呢？  
   0826的接收包里出现过好几个密钥（长度为16个字节嘛）。我们先用第一个16个字节的“密钥”来解一下密：

46 70 36 54 79 6A 70 68 63 4B 63 67 7A 33 4B 46

啊呀，不得了，一不小心被解开了：

[  
00 07 00 88 00 04 52 32 BD 85 2A 30 40 5F 00 00  
00 00 00 78 CB 96 C2 05 20 D4 AC 6A DD 42 81 1A  
D0 A3 CB CC 74 0B 48 E3 CC CC 10 31 0A 22 31 70  
72 23 29 00 68 A8 B8 62 F7 A0 2F E1 B2 A5 A0 4A  
B4 FF 9A 67 C4 AD 62 FC C3 F9 6A 08 37 8E 7F 6D  
7D 4A 42 BA AF 91 DB D9 C9 87 69 34 AF 2E B6 7C  
1E A8 5A 44 F8 0B 14 F3 6E 8A 54 9C 5D 97 96 E0  
6B 7A 81 75 CE 40 04 BB 83 C3 3D F3 37 75 A4 A0  
3C A3 13 AF 55 BF EA D8 43 04 75 E3 00 0C 00 16  
00 02 00 00 00 00 00 00 00 00 00 00 70 5F F2 74  
1F 40 00 00 00 00 00 15 00 30 00 01 01 3E 29 AA  
EE 00 10 EA F8 F4 9C 00 8B 3A C2 55 F7 78 D5 E7  
C5 20 63 02 56 73 44 C4 00 10 AD D9 9C D2 8B 81  
57 4C 35 EC 74 A8 B7 57 0C 9A 00 36 00 12 00 02  
00 01 00 00 00 01 00 00 00 00 00 00 00 00 00 00  
00 18 00 16 00 01 00 00 04 33 00 00 00 01 00 00  
14 77 17 AC C4 B7 00 00 00 00 00 1F 00 22 00 01  
2A F3 14 58 E3 94 F2 74 8F EB F4 86 8C 62 18 66  
E1 B9 5B 26 2A 05 97 17 92 71 44 2C 7E B5 73 29  
01 05 00 CA 00 01 01 03 00 40 02 01 03 3C 01 03  
00 00 81 8B 8A CC 7A 5E 75 FC 4A E7 5A A0 04 AB  
0B B9 BA 19 6C AD 08 08 0B 0D 34 08 A4 C9 27 E9  
AB BC 80 0D 38 97 4B 7F A9 34 99 0B FD C1 14 B4  
97 6A C9 DD 53 23 F1 C3 0D 45 00 40 02 01 03 3C  
01 03 00 00 81 8B 8A CC 7A 5E 75 FC 4A E7 5A A0  
04 AB 0B B9 BA 19 6C AD 08 08 0B 0D 34 08 A4 C9  
27 E9 AB BC 80 0D 38 97 4B 7F A9 34 99 0B FD C1  
14 B4 97 6A C9 DD 53 23 F1 C3 0D 45 00 40 02 02  
03 3C 01 03 00 00 07 EA 88 15 38 C8 52 F3 0E 00  
0F 61 A5 12 DE 2E B8 DE D2 7A FA 46 32 11 21 76  
6C DF 11 96 D7 BB F9 A9 46 41 52 B8 1D 01 22 C0  
AA 18 39 98 93 21 FE 4D 8B 1A 16 C8 31 1D 01 0B  
00 38 00 01 F3 12 1C DC 05 2B 61 49 03 F9 7A 86  
AC 5F 5F 3D D1 10 00 00 00 00 00 00 00 02 00 18  
E1 17 B8 35 88 12 85 44 64 49 EC 16 89 D6 EF 78  
52 5D 44 77 B4 6D C7 96 00 00 00 2D 00 06 00 01  
C0 A8 01 65  
]

解开后还是稍不了多次对比。最后分析如下：

[  
00 07 00 88 00 04   
52 32 BD 85 [ 时间：2013/9/13 15:23:49]   
2A 30 40 5F[ IP地址：42.48.64.95]  
00 00 00 00   
00 78 //从0826接收包里获取得到的令牌  
CB 96 C2 05 20 D4 AC 6A DD 42 81 1A D0 A3 CB CC  
74 0B 48 E3 CC CC 10 31 0A 22 31 70 72 23 29 00  
68 A8 B8 62 F7 A0 2F E1 B2 A5 A0 4A B4 FF 9A 67  
C4 AD 62 FC C3 F9 6A 08 37 8E 7F 6D 7D 4A 42 BA  
AF 91 DB D9 C9 87 69 34 AF 2E B6 7C 1E A8 5A 44  
F8 0B 14 F3 6E 8A 54 9C 5D 97 96 E0 6B 7A 81 75  
CE 40 04 BB 83 C3 3D F3 37 75 A4 A0 3C A3 13 AF  
55 BF EA D8 43 04 75 E3   
00 0C 00 16 00 02 00 00 00 00 00 00 00 00 00 00 //固定  
70 5F F2 74 [ 服务器IP地址：112.95.242.116]  
1F 40 //服务器端口：8000  
00 00 00 00 00 15 00 30 00 01 //固定,标志着0x30个CRC数据的字节开始  
01 //CRC的序号  
3E 29 AA EE //反着的数据，自己慢慢体会。CRC\_Code0  
00 10 //16个字节,CRC\_DATA0  
EA F8 F4 9C 00 8B 3A C2 55 F7 78 D5 E7 C5 20 63  
02 //CRC序号  
56 73 44 C4 //CRC\_code1  
00 10 //16个字节，CRC\_DATA1  
AD D9 9C D2 8B 81 57 4C 35 EC 74 A8 B7 57 0C 9A  
00 36 00 12 00 02 00 01 00 00 00 //固定  
01 00 00 00 00 00 00 00 00 00 00 //固定11个字节，有验证码的时候第一位是13！  
00 18 00 16 00 01 //0825\_fix0  
00 00 04 33 00 00 00 01 00 00 14 77//0825\_fix1  
17 AC C4 B7 [QQ号码：397198519]   
00 00 00 00 00 1F 00 22 00 01//固定  
2A F3 14 58 E3 94 F2 74 8F EB F4 86 8C 62 18 66  
E1 B9 5B 26 2A 05 97 17 92 71 44 2C 7E B5 73 29//32机器码  
01 05 //固定2个字节  
  
00 CA //接下来总共有202个字节  
00 01 01   
03 //标志下面需要3个数据块,这里发现前两个是重复的，有木有？  
00 40   
02 01 03 3C 01 03 00 00 //从01开始，往后的数据序号依次递增  
81 8B 8A CC 7A 5E 75 FC 4A E7 5A A0 04 AB 0B B9  
BA 19 6C AD 08 08 0B 0D 34 08 A4 C9 27 E9 AB BC  
80 0D 38 97 4B 7F A9 34 99 0B FD C1 14 B4 97 6A  
C9 DD 53 23 F1 C3 0D 45  
00 40   
02 01 03 3C 01 03 00 00 //解释同上  
81 8B 8A CC 7A 5E 75 FC 4A E7 5A A0 04 AB 0B B9  
BA 19 6C AD 08 08 0B 0D 34 08 A4 C9 27 E9 AB BC  
80 0D 38 97 4B 7F A9 34 99 0B FD C1 14 B4 97 6A  
C9 DD 53 23 F1 C3 0D 45  
00 40 //  
02 02 03 3C 01 03 00 00 //解释同上  
07 EA 88 15 38 C8 52 F3 0E 00 0F 61 A5 12 DE 2E  
B8 DE D2 7A FA 46 32 11 21 76 6C DF 11 96 D7 BB  
F9 A9 46 41 52 B8 1D 01 22 C0 AA 18 39 98 93 21  
FE 4D 8B 1A 16 C8 31 1D  
01 0B 00 38 00 01 //固定  
F3 12 1C DC 05 2B 61 49 03 F9 7A 86 AC 5F 5F 3D//  
D1 //登录标志  
10 00 00 00 00 00 00 00 02 //9个字节固定  
00 18 //如果前面的发送包错误的话，这里的令牌是取不到的  
E1 17 B8 35 88 12 85 44 64 49 EC 16 89 D6 EF 78  
52 5D 44 77 B4 6D C7 96   
00 00 00 2D 00 06 00 01 //固定，每个版本不一样  
C0 A8 01 65 [ 本地连接的IP地址：192.168.1.101]  
]

好了，0828的发送包都讲完了，革命快要成功了，加油！

## [0828 接收包](http://www.10qf.com/forum.php?mod=viewthread&tid=44&extra=page%3D1)

 在0828的发送包发完之后，如果数据正确的话，基本就会有0828的接收包返回了。如果没返回，毫不犹豫地告诉你，你的发包有问题。

[NO.6 2013-09-13 15:23:33 261 RECEIVE 431字节]  
02 34 27 08 28 6F 0E 17 AC C4 B7 00 00 00 A2 1E  
A8 AC 52 F6 C2 35 83 64 9F 82 A2 9E DB C0 2E 46  
5E 99 00 8B 2A 48 99 4F EF 5A EC A3 6C DA 73 61  
00 D2 79 99 22 75 52 98 81 77 8C 9F C2 F4 23 1B  
47 D2 02 91 A1 70 43 8B 5E 8F A4 97 53 5C 43 A7  
0F CF 2A CE 58 C3 D4 80 55 81 6B FB 52 78 86 C3  
42 C4 75 E0 F1 F7 EA E3 B7 B8 45 C2 A2 4D D3 58  
1E 26 72 3C 6F DF 19 40 47 80 94 FF A7 CF 08 81  
30 3A 2B 67 B4 CC 2E 5E 19 74 CF 3F 38 90 01 99  
0D D6 C0 FC 96 8E 00 8E BA 4B 5F CA 28 82 DA FC  
7F C7 EA 47 20 F8 89 65 5D 2A 70 7A D4 04 B6 57  
A0 FA DE 6A 7B 0C 98 11 EB 8C 3B B0 F5 B0 1C 05  
3A DB 2D 39 C2 A0 B9 5D 60 63 66 A4 D7 EC 82 FC  
52 5E 47 41 9E E2 9A 5F 82 53 B6 56 5B 98 68 0D  
F4 7E 1F 10 02 A1 54 AB BD 46 50 2D AD 2C 0B 7F  
38 AE 2E 59 8C 6C FD 2E 7A 5B 71 27 F2 84 32 E3  
D6 A8 2B 89 79 ED 82 AE 6B AF 2E 86 E9 66 62 57  
CE 98 37 5E F5 47 47 E4 E4 13 F9 83 DA 49 FD 43  
64 C6 B9 47 C0 B5 6D F6 EF 63 4E A8 C0 65 35 DB  
24 6C 54 2A 26 23 AE 57 DC E4 33 02 5A 92 0E 64  
45 E2 D0 86 27 5C 9A ED 24 F3 F9 63 C5 C8 D8 71  
42 9D 4A 93 D8 7A D0 96 3B C7 F3 2E E0 5A 81 1F  
78 47 B1 4D 34 4E A5 D5 D5 C2 73 47 E7 4C 57 10  
DF 2F 86 84 B3 98 22 40 94 79 12 6C BE DF 6F 0E  
5E F1 C3 CA 40 52 39 23 CD C1 10 6A 5A 87 B0 44  
F5 96 7D 74 57 75 40 96 89 F2 76 6D 74 B3 AB 56  
6A 37 13 8D 29 8C 1A 69 FD 3A EB 5C 87 C9 03

接收包的头部跟别的包没有什么变化：

02 34 27 08 28 6F 0E 17 AC C4 B7 00 00 00

那么后面的数据要用什么密钥解密呢？  
之前我们说过，0826的接收包里宝贝很多，16个字节的数据到处都是，我们从第一个开始试，一个一个试下去。试到第三个变解开了。

61 57 37 71 4A 75 76 53 54 56 37 4B 49 50 25 7A //16个字节的密钥，用于解开0828的接收包

因此我们豪迈地说：世界没有解不开的包。

[  
00 00 0D 00 06 00 01 00 00 00 00 00 14 00 04 00  
01 00 3C 01 0C 00 73 00 01 78 61 45 32 6E 70 6B  
45 6D 6B 72 55 35 63 34 4D 17 AC C4 B7 2A 30 40  
5F F2 EE 52 32 BD 84 00 00 00 B4 03 70 5F F2 74  
70 5F F0 9A 00 00 00 07 00 30 D1 E0 28 39 5B 18  
AA C1 4F EC C7 0B 5C 35 1A B2 16 95 32 C7 22 CE  
02 7A FB 01 65 25 76 2B 1F D9 D3 EC 99 39 05 9D  
66 99 DB 12 1A B0 C0 FE 73 19 08 02 04 08 08 08  
08 08 04 00 05 01 70 5F F2 74 01 0E 00 56 00 01  
00 52 13 48 C2 42 00 86 01 00 00 18 E2 80 98 A5  
39 08 E1 17 E6 C1 1E 98 34 8E 7D 07 76 B9 97 C0  
B8 D2 81 6A 00 20 51 1A 44 62 C9 87 96 1B 08 E4  
B5 F6 09 59 19 5E C3 9F 9F 59 31 FA BD 97 02 39  
49 78 52 13 83 E6 00 0C 00 00 00 01 00 00 00 00  
00 00 00 C0 00 2F 00 2A 00 01 8F FE 4F BB B2 63  
C7 69 C3 F1 3C DC A1 E8 77 A3 DD 97 FA 00 36 04  
40 EF 11 7A 31 02 4E 10 13 94 02 28 00 00 00 00  
00 00 01 0D 00 2C 00 01 00 28 EC 41 8B 04 E9 A3  
F1 4A B7 84 2E BC 88 38 4E D7 2D 3F 14 30 0E 5D  
C5 5B E9 83 4B 3A 21 00 15 09 A6 E5 EC C7 46 53  
FE 8A 01 05 00 49 00 01 02 03 00 15 02 00 01 03  
10 26 31 B0 C5 36 4A 53 5B D1 84 C5 24 8A C7 FC  
6B 00 15 02 00 01 03 10 26 31 B0 C5 36 4A 53 5B  
D1 84 C5 24 8A C7 FC 6B 00 15 02 00 02 03 10 4A  
9A 53 64 28 F5 F6 6A B4 53 EB F8 05 F2 5A F3  
]

包都解开了，里面有什么挑什么，都已经这么明了了。

本帖隐藏的内容

[  
00 00 0D 00 06 00 01 00 00 00 00 00 14 00 04 00 //固定  
01 00 3C 01 0C 00 73 00 01 //固定，但是每个版本都有区别  
78 61 45 32 6E 70 6B 45 6D 6B 72 55 35 63 34 4D //SessionKey，也就是会话密钥，到这一步登录就算完成，但是真正看到头像变化还要到心跳包的位置。  
17 AC C4 B7 [QQ号码：397198519]  
2A 30 40 5F[ IP地址：42.48.64.95]  
F2 EE //本机局域网对外端口：62190 //每台机器产生的结果不一定相同  
52 32 BD 84 [ 时间：2013/9/13 15:23:48]   
00 00 00 B4 03 70 5F F2 74 70 5F F0 9A 00 00 00  
07 //固定，每个版本可能不一样  
00 30 //  
D1 E0 28 39 5B 18 AA C1 4F EC C7 0B 5C 35 1A B2  
16 95 32 C7 22 CE 02 7A FB 01 65 25 76 2B 1F D9  
D3 EC 99 39 05 9D 66 99 DB 12 1A B0 C0 FE 73 19  
08 02 04 08 08 08 08 08 04 //固定  
00 05 01 70 5F F2 74 01 0E 00 56 00 01  
00 52 13 48 C2 42 00 86 01 00   
00 18   
E2 80 98 A5 39 08 E1 17 E6 C1 1E 98 34 8E 7D 07  
76 B9 97 C0 B8 D2 81 6A   
00 20   
51 1A 44 62 C9 87 96 1B 08 E4 B5 F6 09 59 19 5E  
C3 9F 9F 59 31 FA BD 97 02 39 49 78 52 13 83 E6  
00 0C 00 00 00 01 00 00 00 00 00 00  
00 C0   
00 2F 00 2A 00 01 8F FE 4F BB B2 63 C7 69 C3 F1  
3C DC A1 E8 77 A3 DD 97 FA 00 36 04 40 EF 11 7A  
31 02 4E 10 13 94 02 28 00 00 00 00 00 00 01 0D  
00 2C 00 01   
00 28   
EC 41 8B 04 E9 A3 F1 4A B7 84 2E BC 88 38 4E D7  
2D 3F 14 30 0E 5D C5 5B E9 83 4B 3A 21 00 15 09  
A6 E5 EC C7 46 53 FE 8A  
01 05 00 49 00 01 02 03 //标志这0x49个字节的数据，这个跟发送包那几个0040的3个数据有关系，看出来了吗?但是具体什么含义，待广大同仁继续努力！  
00 15 02 00 01 03 10   
26 31 B0 C5 36 4A 53 5B D1 84 C5 24 8A C7 FC 6B  
00 15 02 00 01 03 10   
26 31 B0 C5 36 4A 53 5B D1 84 C5 24 8A C7 FC 6B  
00 15 02 00 02 03 10   
4A 9A 53 64 28 F5 F6 6A B4 53 EB F8 05 F2 5A F3  
]

基本到此为止，那些没写注释的，是我也不知道这些代表什么意思。但是目前来说没有能用上这些数据的包。我们只要我们需要的数据就好了，比如会话密钥。  
  
另外后面的那些包都是一些小罗罗，根本不值得一提了，因为基本都是用我们的这个sessionkey解开。  
没什么难度。过两天我们再来解析那些需要验证码的数据包。再见！

## 0826的验证码问题

很久没有更新这个系列了.由于本人工作很忙,一直到今天才抽时间写一下,主要还是看到有这么多朋友支持,让我内牛满面...http://cache.soso.com/img/img/e113.gif  
     在低版本(比如2011,2012等等)里,很多数据都没有与帐号挂钩.意思就是说,即便你把一些令牌之类的数据都固定下来也不影响登录.所有号码通吃.但是2013开始就不行了.很多数据都直接操作帐号了.下面以我遇到的问题给大家分享一下我的看法..  
     这里我们重点看一下0826的验证码问题.  
     0826如果是需要验证码的话,不能用普通的密钥去解密,而是由之前我们讲到的decodekey来解密.而decodekey是用ecc算法得出来的,因此0826的接收包这个时候必须用这个decodekey才能解开.还没有接了0826的可以看这篇文章:[PCQQ UDP协议的研究<4>——0826 发送包](http://www.10qf.com/thread-40-1-1.html)   
     下面请看我测试的例子:  
  
准备一组ecc算出来的decodekey(这组key大家可以记住固定到自己的程序里):

hdKey:   
03 F4 72 8F 4A 78 8C 4F 65 F8 0C 7B D1 BB 54 43 3F DA 70 23 83 C2 74 D2 CD  
DecodeKey:   
C3 FC 26 86 0A B9 67 CE AE 85 76 15 87 3E 5E 28

我用一个需要验证码才能登录的号码在自己程序里测试一下.

[NO.1 2013-10-25 23:29:18 825 SEND 147字节]  
02 34 27 08 25 47 A2 8A E2 DA 2A 03 00 00 00 01  
01 01 00 00 66 68 00 00 00 00 16 BE E6 6A F3 24  
F1 2A 90 F5 46 C1 51 34 53 4A 89 21 5E ED 6B 00  
B1 36 7A 6C 36 5C 7C EC 42 F2 B6 7A A8 8B B3 5F  
6B 81 0A F9 FA 56 57 B2 35 2F 6C 20 0A 97 32 69  
47 02 65 1D 9B A6 E6 DA 54 DA 87 D2 3E 8E 2A B2  
4D C1 A4 46 A5 A7 F4 E3 C0 4E 6D 8C DC 34 28 D3  
E9 FF 0C C6 6B CA 1E 3B 81 51 8B D7 7E EA FA 72  
27 46 AF B4 0F AA 2F 3F D8 74 3C BB 09 E7 CC B0  
AD 6E   
[  
00 18 00 16 00 01 00 00 04 33 00 00 00 01 00 00  
14 77 8A E2 DA 2A 00 00 00 00 03 09 00 08 00 01  
70 5F F0 CA 00 02 00 36 00 12 00 02 00 01 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 01 14 00 1D  
01 02   
00 19 //这个都固定,对应的decodekey在后面可以到,我添加上去的  
03 F4 72 8F 4A 78 8C 4F 65 F8 0C 7B D1 BB 54 43  
3F DA 70 23 83 C2 74 D2 CD  
]  
03  
  
[NO.2 2013-10-25 23:29:18 974 RECEIVE 111字节]  
02 34 27 08 25 47 A2 8A E2 DA 2A 00 00 00 CD BB  
40 E5 F9 12 0F F5 B7 98 FD 1D B8 BD 19 8F F8 3D  
05 80 F0 D0 F9 F7 D1 BD BC C8 CA 51 10 E5 96 21  
AD DE D4 20 7D 06 95 97 DC 91 2F 4B 92 31 2D 3E  
91 FB 44 2A D8 B7 08 54 07 FB 84 2E D2 5E 06 88  
19 D1 85 56 A7 B9 4C 3C 64 B8 04 99 A8 9C A1 04  
43 D0 95 D2 5F DD 4C 74 BC F3 59 58 44 D9 03  
  
[NO.3 2013-10-25 23:29:19 019 SEND 603字节]  
02 34 27 08 26 C3 27 8A E2 DA 2A 03 00 00 00 01  
01 01 00 00 66 68 00 00 00 00 3B 7A A5 EB CD 4D  
B9 E0 25 9E 1D 03 AC 92 5E 87 13 FA 13 42 C1 08  
7D 20 A2 F3 B2 21 18 06 4A CC 3F 4F 8C 6E 95 3E  
43 99 7F B0 AB 7C A1 DB F5 3A F7 7C DD 4E AD 0E  
46 48 69 D9 22 DC CC 35 06 9D 48 3A D0 91 95 62  
8A CA 2A C6 7D B2 EC D7 02 40 4B FF 2C F0 42 17  
27 96 4A D2 8E B3 0D AF 89 FA 0D 7C D1 28 B2 54  
70 54 2C 26 B3 D5 FC 2F F4 6F 4B 75 AA 19 85 BB  
41 21 76 97 18 A9 44 17 26 AE 7E AD C9 A8 0C 18  
35 0E 12 56 AF D1 5A 04 8A 2A BB 45 49 8A A0 AD  
4E 37 6D 89 AC E8 91 29 9C AB 68 05 E4 76 10 66  
A7 1E 34 AE 14 5A 55 ED 99 F1 23 08 F4 02 3E EF  
0F DF 81 DC 16 DC 7E 81 D0 39 9E 87 51 43 F5 2A  
7B 0F 26 D1 1A 0E 60 22 24 4C C2 26 3B D5 6E D1  
1D 53 7C 6F 81 F1 2E 32 D8 02 E7 E3 0E 9A 97 1B  
AD A4 FB 80 BF CD 3A D8 02 86 20 14 58 93 F8 64  
A9 72 6C F6 C0 68 68 A0 FB 68 95 EA 28 00 6C 13  
37 7D 41 C3 FF 73 B9 71 18 0C 2A A7 11 A3 D5 9A  
A9 B7 55 96 D3 1E 4E 52 A7 85 B2 57 AE D3 CE FA  
7B 32 2D 5B D8 94 87 BA 7A 01 E9 E4 F3 9E A5 86  
91 9B 36 D5 FD D6 D0 1E 55 B3 CF 01 0E 30 D0 2B  
09 C7 55 B8 AA A8 D6 7E 5A 5B 55 E2 6E 92 F7 F8  
1B 9A AE 00 A2 D3 81 C3 BC 34 98 76 FF 90 A0 BF  
50 EB 42 47 38 BE 42 94 7D 8E 7C 49 4E 71 74 00  
E3 7E 05 04 88 2C 5A A7 43 42 C6 27 D7 90 D0 C8  
B1 9E 9B 81 DF 5A 14 40 DB 29 D6 42 A7 2A 78 03  
05 E6 BC 9C 19 22 9B DC F4 19 DB E0 2C F2 94 8E  
3B 9B D0 C7 74 6D 07 D7 E3 45 F8 35 7A A2 2A 8D  
C9 99 BE 45 F0 85 98 DC E3 14 C3 2E ED A3 14 16  
69 4D 36 B5 98 45 EC F0 4A 43 62 09 B1 88 52 0C  
2A 04 BE 6B 77 05 20 A2 1F B3 5F 46 50 CC 4E 80  
6B 35 7A 11 91 EA A9 C2 B6 FD E6 07 41 C2 B2 FA  
DD 5B A3 F1 AF 3A 75 04 52 62 7B 8D 44 C8 86 E5  
44 5F F9 EF 59 AF 0F 41 48 FC 34 03 47 C0 E7 3E  
48 29 57 5B 5E 1B 4F 50 4A 2C 7B 00 52 39 7B 26  
21 AA 16 1F B3 A6 9F EE 29 BD EF 04 3C 0D 10 F1  
54 23 E7 87 94 17 17 ED 21 F9 03  
  
  
[NO.4 2013-10-25 23:29:19 179 RECEIVE 871字节]  
02 34 27 08 26 C3 27 8A E2 DA 2A 00 00 00 27 37  
50 C0 64 0D 79 9F D7 3F A8 79 D1 52 F9 EF 10 01  
E6 B3 E2 E4 FC 42 79 E9 0F 92 9B 65 2D 9F 65 8E  
39 AB 3C BF 9F 04 C3 51 AB F1 5D C0 46 AC 60 05  
D1 B6 BB D7 A8 60 17 30 F1 33 F9 70 76 A5 9B 3C  
68 5A FB 49 E3 DB A0 01 F8 E4 36 5E 22 C5 03 6E  
4D 84 0C 68 7A BD AC 7C 71 35 DF 6E E4 A4 37 79  
C4 64 6C 19 7A 2E 58 29 EF 20 10 04 58 22 BB 58  
13 F1 4F 7A 5F 43 1E E2 AF 91 1E 92 AC 9D 19 AE  
67 31 CC 2E 80 62 6D 15 D3 C6 BA FE A4 F7 3D D4  
B0 D7 91 47 6B A5 70 5F 5D 65 71 2A BE 8F A9 08  
AF 73 7C 60 ED 77 D1 17 9C E1 E9 4C 07 37 B1 14  
59 E7 FF E7 F8 9B A8 85 BC B7 7D A7 AB 2D D9 0D  
14 3D 39 8D A5 BD 26 AB 0D 6A D0 CC 8F 9D C0 C2  
6E 7E 76 93 D5 D3 D3 96 DD 01 9A C8 06 D1 23 86  
23 86 FF 12 26 2C 32 CF 31 CF A7 EF AD 2B 12 3B  
85 AD D3 C3 3B DC F7 A6 35 2C BC A3 77 68 FA 72  
1F 01 D1 86 7E CC E2 66 4E AD F2 1E 31 F0 32 9A  
C8 B4 67 9A DB 53 43 0D BB 02 7F 1B 69 B3 7C F1  
FF 2F 72 AF D6 14 12 9D 28 5E 21 8B 0A CD B0 ED  
8C 9E 77 22 56 96 50 B9 E3 7B 44 61 CE EF EF 1C  
FD BF 7F FC 37 73 8D 4F 03 CD C1 EF 06 30 E5 CD  
DC 35 7D 0C 66 1B 1B 06 98 F7 27 81 87 24 80 E5  
3D 67 63 EC AC 3F 1C D5 45 0A 6A 8E 83 42 BF 84  
4A 39 9C E0 17 BF 02 48 2E 6E 5A AD 33 7F E6 23  
91 AC F4 5C 5A EE 2B 16 29 75 F9 FA CC F7 69 E2  
15 08 30 53 8A 37 3E 84 14 BD F6 12 54 D4 B6 5F  
9E 5C C1 D0 4E 5F D7 31 E4 EE 32 41 4C 84 CF 79  
6E E7 9B E6 11 D5 89 8F F2 FD A3 C2 8D 74 D8 4B  
0E E1 82 94 00 7E D4 8E E9 BE CB C5 74 C2 32 A2  
0B B1 02 2D 9E FB D2 2B 74 05 0C AF FE B6 4C A9  
63 A0 A8 32 BA 08 4D 9A F9 87 7B 62 4C 7D 35 0A  
50 13 F0 E0 3E 11 7B 0A E2 62 6B 59 89 36 40 3E  
12 06 C8 FD BD 87 FC 93 E2 04 8B 28 90 61 8E 6B  
4D B2 CB 67 FB 58 10 9E 80 8A C5 5F A8 EC A1 09  
C8 23 02 3F DA E7 66 25 B4 A2 4B 3A B4 8D 20 15  
8F A7 90 16 16 7C 37 18 9A 7F D3 5A 03 87 36 63  
E0 C6 B0 69 E2 E7 23 F9 91 E5 89 79 7B AA 61 F0  
33 56 C5 88 FD 7A 7D F4 EC 54 4C 6D 4F E9 AD 8E  
75 16 12 06 2D 03 E2 74 2D 04 16 40 5A 88 2A BF  
81 95 C3 D6 96 D2 CF A3 C7 5A 1C C4 6E 24 13 F0  
A3 32 3B AA 05 7A 1F C0 67 B0 60 70 9C C0 A9 4A  
A9 3A 06 2B 7F 3F 53 4C A1 F8 17 F9 A1 EE BE AF  
5A B6 7D AA 74 25 7E 08 7D 18 55 DA 5C 33 35 A1  
03 76 10 6F 99 B2 5B CA B2 BA 25 CF 30 9F 4B 83  
2B 78 9C 26 05 A8 16 31 B4 DA EE D4 A9 9D 9B 6D  
A5 0A C5 C2 5D 15 8B 97 CE BA 78 DF E3 35 75 4A  
4C 16 8B C8 EB BD A6 67 32 69 1D 8E 37 FD D1 D9  
2C 74 6D 0C 68 E4 E6 8D 13 4E 67 45 C7 EF BD B1  
12 67 69 71 E5 F7 DD 48 17 54 D1 6F 95 F3 6F 8F  
50 68 58 46 1D 8A 28 47 FB 5B C7 DD DD 5C 29 99  
DB 57 58 85 53 13 D4 FD B1 EA E4 CA 61 28 17 EF  
BA B4 9B 08 60 65 20 3A 06 1B E0 EB 6E 3C 7E C6  
8A C0 FE E0 2D 84 7C FF DA 39 AD EC ED C4 A9 35  
BD 49 E6 42 36 EB   
C3 FC 26 86 0A B9 67 CE AE 85 76 15 87 3E 5E 28 //这个是我后面为了解密方便添加进去的.0826decodekey密钥  
  
[  
FB 01 04 03 33 00 01 00 BA 02 03 2C 13 00 05 01  
00 00 01 23 00 38 AA DC 12 F7 23 6F 77 EE B3 4C  
97 84 B4 23 5D 6B 41 C7 F4 58 64 8C 57 43 F8 79  
EF 42 DB 55 8B 24 99 D9 93 64 28 73 20 BC 67 0A  
F1 87 9B DE 34 35 B4 EB DC 39 1C 07 97 67 02 BC  
89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  
00 00 00 82 00 00 00 35 08 03 00 00 00 BA 12 C3  
02 00 00 00 42 50 4C 54 45 FF FF FF EF EF EF DF  
DF DF BF BF BF 6F 6F 6F 2F 2F 2F 0F 0F 0F 5F 5F  
5F 1F 1F 1F 9F 9F 9F 4F 4F 4F 3F 3F 3F 7F 7F 7F  
AF AF AF CF CF CF 8F 8F 8F FF FF FF FF FF FF FF  
FF FF FF FF FF FF FF FF FF FF FF D6 3C DC 5E 00  
00 03 CC 49 44 41 54 58 85 ED 57 D9 96 DC 38 08  
B5 16 10 5A 2C 25 33 93 FF FF D5 00 92 D7 AE 3E  
5D 76 4D 32 67 72 8A 87 8E CB 91 04 BE C0 E5 6A  
9A DE F6 B6 FF A9 19 6B 9E 5E EB 3C 20 86 E7 D7  
3F 63 D6 13 22 92 7F EA D4 C8 6B 93 CF 98 EF 7A  
73 A5 CC A7 57 26 A0 5A 22 3C FF D7 03 0B 08 BA  
CA E3 4D 18 3C 12 9D 40 AC 82 00 12 A4 69 A2 E4  
BE 3A 20 62 5A 4E BA 17 81 41 CF B0 17 A2 2D 86  
CA FE 63 D4 DF 96 E3 FB 2A 86 80 76 9C 94 EE 85  
E0 B0 75 5F 65 79 63 19 83 B8 FC 98 19 A0 E3 86  
1F 63 55 C9 48 41 9C 97 81 7F 79 26 69 0F 0D 40  
FF C9 AB A7 84 60 D7 00 B9 28 DA 83 4D 36 21 66  
1F 40 00 A8 82 23 1B D1 CD 08 A6 49 33 68 56 4F  
8D 34 82 DA 42 D6 92 58 D0 A9 76 DB 52 91 9A 7E  
3A 48 13 80 C2 10 1F C6 7A C1 E2 48 28 C7 22 59  
70 EC 9D 90 3F 35 2D AF B9 EF 71 6D 51 A0 F1 BA  
A0 93 62 90 F7 F9 3E 08 C3 96 9E 6E 28 A8 B0 3F  
88 B6 81 33 CE A9 DF 84 BE 85 15 92 15 9A 59 B8  
40 33 60 EF 82 F0 E3 DE 36 9A 16 3C B8 70 92 06  
D4 EE 92 C2 B4 4B C1 35 5B 92 D2 08 B5 9E E1 36  
35 F2 77 C8 DE 17 F6 4F 55 FE 98 AD A9 2F 9B 91  
EA B3 98 E7 4C 94 8B E3 6A 44 3E D3 A4 09 42 80  
CE 4A 86 92 9F 71 30 D4 27 90 C5 DB A4 30 72 18  
A5 FB 42 E3 D1 24 7D 28 24 57 33 9A D5 5B 49 38  
AF D5 06 4C 20 DF 3F 1C E3 5E 68 C9 2C 0E 3D 0E  
3A 72 49 7A 6F A3 E4 39 C8 0F 86 26 75 92 34 85  
94 39 EB 5C 6F 7B 3C 9B D5 13 23 E6 A5 9E A5 29  
73 8F 21 82 04 24 DC E7 07 45 09 5D 94 1E 32 52  
31 4B 0B BF 66 1F 7B 09 18 74 F4 73 54 C1 90 32  
7A AE 02 CB 8F 92 6B 02 1D 1E 06 69 9E 83 72 58  
DC C6 C9 5D 83 31 DD FE 3E BE EE 72 C1 5B 69 18  
CF 30 50 2F D2 E1 AF 4F 05 1B 18 3B B3 0B C1 30  
68 E1 72 87 DB 17 0A 79 6F 03 4A 99 6A 57 83 28  
5B 2A F9 13 20 87 0F 02 EA 6A 14 4C 55 D7 34 24  
AC 4F 0D D3 71 67 4D 40 DC 05 26 73 D9 09 5B 30  
E0 61 4C 22 BB AA C4 82 30 9D 2D 5C E8 CF 83 CB  
88 07 75 24 82 20 78 E0 AA 0B 49 BA 00 01 00 28  
BA 0D 07 F4 4A 09 88 5D 7B EB 0C 70 05 E1 08 DA  
52 87 97 63 72 95 F0 4D 99 DD 06 CA DC 17 6D DB  
19 DA B0 80 CA 22 E8 71 01 15 00 10 E4 F3 9C FA  
CA 25 03 5B 38 18 F3 C6 8D BA 2C 41  
]  
03

所以看到这里我想各位都知道怎么做了吧.!非常感谢各位捧场.希望能让各位找到自己所需的资源.

## PCQQ加密类Tea算法[最新版本兼容]

QQ使用的TEA虽然是标准的TEA，但是QQ在使用这个算法的时候，由于需要加密不定长的数据，所以使用了一些常规的填充办法和交织算法（也就是说，把前一组的加密结果和后一组未加密的结果进行运算，产生新的结果）。QQ消 息被分为多个加密单元，每一个加密单元都是8字节，使用TEA进行加密，加密结果再作为下一个单元 的密钥。如果明文本身的长度不是8的倍数，那么还要进行填充，使其成为8的倍数。填充的时候会用一 个32位随机数存放于明文的开始位置，再在明文的最后用0填充为整个长度是8的 倍数。由于会向后反馈，这样即使对于相同的明文,因为使用了不同的随机数，也会产生完全不同的密文。使用这种特殊的填充反馈算 法所导致的结果就是，一段密文只能用加密它的密钥进行解密，如果使用不正确的密钥，就无法得到正确的填充结果。最常见的就是解密后得到的填充数值不是0， 这样就判断解密失败。

QQ消息的加密算法是一个16次的迭代过程，并且是反馈的，每一个加密单元是8字节，输出也是8字节，密钥是16字节

以prePlain表示前一个明文块，plain表示当前明文块，crypt表 示当前明文块加密得到的密文块，preCrypt表示前一个密文块

f表示加密算法， 那么从plain得到crypt的 过程是: crypt = f(plain ˆ preCrypt ) &circ

d表示解密算法，从crypt得到plain的过程 自然是 plain = d(crypt ˆ prePlain) ˆ

填充机制，其会在明文前和明文后分别填充一定的字节数，以保证明文长度是8字节的倍数填充的字节数与原始明 文长度有关，填充的方法是:

------- 消息填充算法 -----------

a = (明文长度 + 10) mod 8 //计算填充长度

if(a 不等于 0) a = 8 - a;

b = 随机数 & 0xF8 | a; //这个的作用是把a的值保存了下来

plain[0] = b; //然后把b做为明文的第0个字节，这样第0个 字节就保存了a的信息，这个信息在解密时就要用来找到真正明文的起始位置

plain[1 至 a+2] = 随机 数 & 0xFF; // 这里用随机数填充明文的第1到 第a+2个字节

plain[a+3 至 a+3+明文长度-1] = 明文; //从a+3字 节开始才是真正的明文

plain[a+3+明文长度, 最后] = 0; //在最后，填充0，填充到总长度为8的 整数为止。到此为止，结束了，这就是最后得到的要加密的明文内容

下面给大伙贴出.net版本的QQ Tea算法：

本帖隐藏的内容

附 源码，见文章结尾

using System;

using System.Collections.Generic;

using System.Text;

using System.Security.Cryptography;

namespace Lingchen.Common.Utils

{

public class TeaCrypter

{

static MD5 MD5Instance = System.Security.Cryptography.MD5.Create();

private byte[] Plain; //指向当前的明文块

private byte[] prePlain; //指向前面一个明文块

private byte[] Out; //输出的密文或者明文

private long Crypt, preCrypt; //当前加密的密文位置和上一次加密的密文块位置，他们相差8

private long Pos; //当前处理的加密解密块的位置

private long padding; //填充数

private byte[] Key = new byte[16]; //密钥

private bool Header; //用于加密时，表示当前是否是第一个8字节块，因为加密算法

//是反馈的，但是最开始的8个字节没有反馈可用，所有需要标

//明这种情况

private long contextStart; //这个表示当前解密开始的位置，之所以要这么一个变量是为了

//避免当解密到最后时后面已经没有数据，这时候就会出错，这

//个变量就是用来判断这种情况免得出错

public TeaCrypter()

{

//

// TODO: 在此处添加构造函数逻辑

//

}

/// <summary>

/// MD5加密

/// </summary>

/// <param name="data">The data.</param>

/// <returns></returns>

public static byte[] MD5(byte[] data)

{

return MD5Instance.ComputeHash(data);

}

//Push 数据

byte[] CopyMemory(byte[] arr, int arr\_index, long input) //lenth = 4

{

if (arr\_index + 4 > arr.Length)

{

// 不能执行

return arr;

}

arr[arr\_index + 3] = (byte)((input & 0xff000000) >> 24);

arr[arr\_index + 2] = (byte)((input & 0x00ff0000) >> 16);

arr[arr\_index + 1] = (byte)((input & 0x0000ff00) >> 8);

arr[arr\_index] = (byte)(input & 0x000000ff);

arr[arr\_index] &= 0xff;

arr[arr\_index + 1] &= 0xff;

arr[arr\_index + 2] &= 0xff;

arr[arr\_index + 3] &= 0xff;

return arr;

}

long CopyMemory(long Out, byte[] arr, int arr\_index)

{

if (arr\_index + 4 > arr.Length)

{

return Out;

//不能执行

}

long x1 = arr[arr\_index + 3] << 24;

long x2 = arr[arr\_index + 2] << 16;

long x3 = arr[arr\_index + 1] << 8;

long x4 = arr[arr\_index];

long o = x1 | x2 | x3 | x4;

o &= 0xffffffff;

return o;

}

long getUnsignedInt(byte[] arrayIn, int offset, int len /\*\*//\*Default is 4\*/)

{

long ret = 0;

int end = 0;

if (len > 8)

end = offset + 8;

else

end = offset + len;

for (int i = offset; i < end; i++)

{

ret <<= 8;

ret |= (ushort)(arrayIn & 0xff);

}

return (ret & 0xffffffff) | (ret >> 32);

}

long Rand()

{

Random rd = new Random();

long ret;

ret = rd.Next() + (rd.Next() % 1024);

return ret;

}

private byte[] Decipher(byte[] arrayIn, byte[] arrayKey, long offset)

{

//long Y,z,a,b,c,d;

long sum, delta;

//Y=z=a=b=c=d=0;

byte[] tmpArray = new byte[24];

byte[] tmpOut = new byte[8];

if (arrayIn.Length < 8)

{

// Error:return

return tmpOut;

}

if (arrayKey.Length < 16)

{

// Error:return

return tmpOut;

}

sum = 0xE3779B90;

sum = sum & 0xFFFFFFFF;

delta = 0x9E3779B9;

delta = delta & 0xFFFFFFFF;

long Y = getUnsignedInt(arrayIn, (int)offset, 4);

long z = getUnsignedInt(arrayIn, (int)offset + 4, 4);

long a = getUnsignedInt(arrayKey, 0, 4);

long b = getUnsignedInt(arrayKey, 4, 4);

long c = getUnsignedInt(arrayKey, 8, 4);

long d = getUnsignedInt(arrayKey, 12, 4);

for (int i = 1; i <= 16; i++)

{

z -= ((Y << 4) + c) ^ (Y + sum) ^ ((Y >> 5) + d);

z &= 0xFFFFFFFF;

Y -= ((z << 4) + a) ^ (z + sum) ^ ((z >> 5) + b);

Y &= 0xFFFFFFFF;

sum -= delta;

sum &= 0xFFFFFFFF;

}

tmpArray = CopyMemory(tmpArray, 0, Y);

tmpArray = CopyMemory(tmpArray, 4, z);

tmpOut[0] = tmpArray[3];

tmpOut[1] = tmpArray[2];

tmpOut[2] = tmpArray[1];

tmpOut[3] = tmpArray[0];

tmpOut[4] = tmpArray[7];

tmpOut[5] = tmpArray[6];

tmpOut[6] = tmpArray[5];

tmpOut[7] = tmpArray[4];

return tmpOut;

}

private byte[] Decipher(byte[] arrayIn, byte[] arrayKey)

{

return Decipher(arrayIn, arrayKey, 0);

}

private byte[] Encipher(byte[] arrayIn, byte[] arrayKey, long offset)

{

byte[] tmpOut = new byte[8];

byte[] tmpArray = new byte[24];

long sum, delta;

if (arrayIn.Length < 8)

{

// Error:

return tmpOut;

}

if (arrayKey.Length < 16)

{

// Error:

return tmpOut;

}

sum = 0;

delta = 0x9E3779B9;

delta &= 0xFFFFFFFF;

long Y = getUnsignedInt(arrayIn, (int)offset, 4);

long z = getUnsignedInt(arrayIn, (int)offset + 4, 4);

long a = getUnsignedInt(arrayKey, 0, 4);

long b = getUnsignedInt(arrayKey, 4, 4);

long c = getUnsignedInt(arrayKey, 8, 4);

long d = getUnsignedInt(arrayKey, 12, 4);

for (int i = 1; i <= 16; i++)

{

sum += delta;

sum &= 0xFFFFFFFF;

Y += ((z << 4) + a) ^ (z + sum) ^ ((z >> 5) + b);

Y &= 0xFFFFFFFF;

z += ((Y << 4) + c) ^ (Y + sum) ^ ((Y >> 5) + d);

z &= 0xFFFFFFFF;

}

tmpArray = CopyMemory(tmpArray, 0, Y);

tmpArray = CopyMemory(tmpArray, 4, z);

tmpOut[0] = tmpArray[3];

tmpOut[1] = tmpArray[2];

tmpOut[2] = tmpArray[1];

tmpOut[3] = tmpArray[0];

tmpOut[4] = tmpArray[7];

tmpOut[5] = tmpArray[6];

tmpOut[6] = tmpArray[5];

tmpOut[7] = tmpArray[4];

return tmpOut;

}

private byte[] Encipher(byte[] arrayIn, byte[] arrayKey)

{

return Encipher(arrayIn, arrayKey, 0);

}

private void Encrypt8Bytes()

{

byte[] Crypted;

for (Pos = 0; Pos <= 7; Pos++)

{

if (this.Header == true)

{

Plain[Pos] = (byte)(Plain[Pos] ^ prePlain[Pos]);

}

else

{

Plain[Pos] = (byte)(Plain[Pos] ^ Out[preCrypt + Pos]);

}

}

Crypted = Encipher(Plain, Key);

for (int i = 0; i <= 7; i++)

{

Out[Crypt + i] = (byte)Crypted;

}

for (Pos = 0; Pos <= 7; Pos++)

{

Out[Crypt + Pos] = (byte)(Out[Crypt + Pos] ^ prePlain[Pos]);

}

Plain.CopyTo(prePlain, 0);

preCrypt = Crypt;

Crypt = Crypt + 8;

Pos = 0;

Header = false;

}

private bool Decrypt8Bytes(byte[] arrayIn, long offset)

{

long lngTemp;

for (Pos = 0; Pos <= 7; Pos++)

{

if (this.contextStart + Pos > arrayIn.Length - 1)

{

return true;

}

prePlain[Pos] = (byte)(prePlain[Pos] ^ arrayIn[offset + Crypt + Pos]);

}

try

{

prePlain = this.Decipher(prePlain, Key);

}

catch

{

return false;

}

lngTemp = prePlain.Length - 1;

contextStart += 8;

Crypt += 8;

Pos = 0;

return true;

}

private bool Decrypt8Bytes(byte[] arrayIn)

{

return Decrypt8Bytes(arrayIn, 0);

}

#region Public Methods!

/\*\*/

/// <summary>

/// QQ TEA 加密函数

/// </summary>

/// <param name="arrayIn">要加密的字串</param>

/// <param name="arrayKey">密钥</param>

/// <param name="offset">偏移</param>

/// <returns></returns>

public byte[] QQ\_Encrypt(byte[] arrayIn, byte[] arrayKey, long offset)

{

Plain = new byte[8];

prePlain = new byte[8];

long l;

Pos = 1;

padding = 0;

Crypt = preCrypt = 0;

arrayKey.CopyTo(Key, 0); // Key Must Be 16 Length!

Header = true;

Pos = 2;

//计算头部填充字节数

Pos = (arrayIn.Length + 10) % 8;

if (Pos != 0)

Pos = 8 - Pos;

//输出长度

Out = new byte[arrayIn.Length + Pos + 10];

//把POS存到PLAIN的第一个字节

//0xf8后面3位是空的，正好给Pos

Plain[0] = (byte)((Rand() & 0xf8) | Pos);

//用随机数填充1到Pos的内容

for (int i = 1; i <= Pos; i++)

{

Plain = (byte)(Rand() & 0xff);

}

Pos++;

padding = 1;

//继续填充两个字节随机数，满8字节就加密

while (padding < 3)

{

if (Pos < 8)

{

Plain[Pos] = (byte)(Rand() & 0xff);

padding++;

Pos++;

}

else if (Pos == 8)

{

this.Encrypt8Bytes();

}

}

int I = (int)offset;

l = 0;

//明文内容，满8字节加密到读完

l = arrayIn.Length;

while (l > 0)

{

if (Pos < 8)

{

Plain[Pos] = arrayIn[I];

I++;

Pos++;

l--;

}

else if (Pos == 8)

{

this.Encrypt8Bytes();

}

}

//末尾填充0，保证是8的倍数

padding = 1;

while (padding < 9)

{

if (Pos < 8)

{

Plain[Pos] = 0;

Pos++;

padding++;

}

else if (Pos == 8)

{

this.Encrypt8Bytes();

}

}

return Out;

}

/// <summary>

/// 解密

/// </summary>

/// <param name="inData">密文</param>

/// <param name="offset">密文开始的位置</param>

/// <param name="len">密文长度</param>

/// <param name="k">密钥</param>

/// <returns>明文</returns>

public byte[] Decrypt(byte[] inData, int offset, int len, byte[] k)

{

return QQ\_Decrypt(inData, k);

}

/// <summary>

/// 需要被解密的密文

/// </summary>

/// <param name="inData">密文</param>

/// <param name="k">密钥</param>

/// <returns>已解密的消息</returns>

public byte[] Decrypt(byte[] inData, byte[] k)

{

return Decrypt(inData, 0, inData.Length, k);

}

public byte[] Encrypt(byte[] arrayIn, byte[] arrayKey)

{

return QQ\_Encrypt(arrayIn, arrayKey, 0);

}

/\*\*/

/// <summary>

/// QQ TEA 解密函数

/// </summary>

/// <param name="arrayIn">要解密字串</param>

/// <param name="arrayKey">密钥</param>

/// <param name="offset">偏移</param>

/// <returns></returns>

public byte[] QQ\_Decrypt(byte[] arrayIn, byte[] arrayKey, long offset)

{

byte[] error = new byte[0];

//检查是否是8的倍数至少16字节

if (arrayIn.Length < 16 || (arrayIn.Length % 8 != 0))

{

//Return What?

return error;

}

if (arrayKey.Length != 16)

{

//Return What?

return error;

}

byte[] m;

long I, Count;

m = new byte[offset + 8];

arrayKey.CopyTo(Key, 0);

Crypt = preCrypt = 0;

//计算消息头部，明文开始的偏移，解密第一字节和7相与得到

prePlain = this.Decipher(arrayIn, arrayKey, offset);

Pos = prePlain[0] & 7;

//计算明文长度

Count = arrayIn.Length - Pos - 10;

if (Count <= 0)

{

//Return What?

return error;

}

Out = new byte[Count];

preCrypt = 0;

Crypt = 8;

this.contextStart = 8;

Pos++;

padding = 1;

//跳过头部

while (padding < 3)

{

if (Pos < 8)

{

Pos++;

padding++;

}

else if (Pos == 8)

{

for (int i = 0; i < m.Length; i++)

m = arrayIn;

if (this.Decrypt8Bytes(arrayIn, offset) == false)

{

//Return What?

return error;

}

}

}

//解密明文

I = 0;

while (Count != 0)

{

if (Pos < 8)

{

Out[I] = (byte)(m[offset + preCrypt + Pos] ^ prePlain[Pos]);

I++;

Count--;

Pos++;

}

else if (Pos == 8)

{

m = arrayIn;

preCrypt = Crypt - 8;

if (this.Decrypt8Bytes(arrayIn, offset) == false)

{

//Return What?

return error;

}

}

}

//最后的解密部分，检查尾部是不是0

for (padding = 1; padding <= 7; padding++)

{

if (Pos < 8)

{

if ((m[offset + preCrypt + Pos] ^ prePlain[Pos]) != 0)

{

//Return What?

return error;

}

Pos++;

}

else if (Pos == 8)

{

for (int i = 0; i < m.Length; i++)

m = arrayIn;

preCrypt = Crypt;

if (this.Decrypt8Bytes(arrayIn, offset) == false)

{

//Return What?

return error;

}

}

}

return Out;

}

/// <summary>

/// 加密

/// </summary>

/// <param name="inData">明文字节数组</param>

/// <param name="offset">开始加密的偏移</param>

/// <param name="len">加密长度</param>

/// <param name="k">密钥</param>

/// <returns>密文字节数组</returns>

public byte[] Encrypt(byte[] inData, int offset, int len, byte[] k)

{

return QQ\_Decrypt(inData, k);

}

public byte[] QQ\_Decrypt(byte[] arrayIn, byte[] arrayKey)

{

return QQ\_Decrypt(arrayIn, arrayKey, 0);

}

#endregion

}

}

## QQ注册涉及到的rsa加密算法[.net]

 QQ注册这一块是比较恼火的,它让我们知道，模拟Http过程不一定会一帆风顺。这个差不多让89%的人难住的协议，到底有哪些难的地方，这里我们不做累赘发言，我们要讨论的只是他的密码传输方式的实现。以下贴出RSA加密算法，net实现  
附上调试过程中遇到的部分返回代码含义及可能的原因：  
  
case 0: 注册成功;  
  
case 1: EMAIL注册成功;  
  
case 2: 验证码错误;  
  
case 4:case 5:case 6: 生日或省份错误;  
  
case 8:case 9: EMAIL错误;  
  
case 13:case 15:昵称错误;  
  
case 20: 需要手机短信验证;  
原因：可能是COOKIE错误  
  
case 21: 恶意注册，暂时禁止;  
原因：注册的账号过多，或RSA算法错误  
  
case 26: 需要手机激活;  
原因：相同IP注册过多或提交的COOKIE已经过期，或SESSION超时  
  
case 30: 浏览器不兼容;  
  
{"ec":2,"em":"vc check error"} 验证码错误  
' {"ec":20,"em":"need sms verify"}手机验证  
' “{"ec":4,"em":"param error"}” ‘参数错误  
' {"ec":21,"em":"reg forbid"}” 很抱歉，您所在的IP存在大量异常注册行为  
' {"ec":26,"em":"need upsms verify"} 手机验证  
' {"ec":20,"em":"need sms verify"} 短信认证  
' {"ec":1,"em":"server busy"} ‘服务繁忙  
' {"ec":9,"em":"email bind error"} ’邮箱已注册

public class RSACrypter

{

private static Random rand = new Random();

private BigInteger n = null;

private BigInteger e = null;

private BigInteger pkcs1pad2(string a, int b)

{

if (b < a.Length + 11)

{

throw new Exception("Message too long for RSA");

}

byte[] c = new byte[b];

int d = a.Length - 1;

while (d >= 0 && b > 0)

{

int e = (int)a[d--];

if (e < 128)

{

c[--b] = Convert.ToByte(e);

}

else if ((e > 127) && (e < 2048))

{

c[--b] = Convert.ToByte(((e & 63) | 128));

c[--b] = Convert.ToByte((e >> 6) | 192);

}

else

{

c[--b] = Convert.ToByte((e & 63) | 128);

c[--b] = Convert.ToByte(((e >> 6) & 63) | 128);

c[--b] = Convert.ToByte(((e >> 12) | 224));

}

}

c[--b] = Convert.ToByte(0);

byte[] temp = new byte[1];

while (b > 2)

{

temp[0] = Convert.ToByte(0);

while (temp[0] == 0)

rand.NextBytes(temp);

c[--b] = temp[0];

}

c[--b] = 2;

c[--b] = 0;

return new BigInteger(c);

}

public void RSASetPublic(string a, string b)

{

if (string.IsNullOrEmpty(a) || string.IsNullOrEmpty(b))

{

throw new Exception("Message too long for RSA");

}

n = new BigInteger(a, 16);

e = new BigInteger(b, 16);

}

private BigInteger RSADoPublic(BigInteger x)

{

return x.modPow(e, n);

}

public string RSAEncrypt(string a)

{

BigInteger tmp = pkcs1pad2(a, (n.bitCount() + 7) >> 3);

tmp = RSADoPublic(tmp);

string result = tmp.ToHexString();

return 0 == (result.Length & 1) ? result : "0" + result;

}

public static string Encrypt(string password)

{

RSACrypter crypter = new RSACrypter();

crypter.RSASetPublic("C4D23C2DB0ECC904FE0CD0CBBCDC988C039D79E1BDA8ED4BFD4D43754EC9693460D15271AB43A59AD6D0F0EEE95424F70920F2C4A08DFDF03661300047CA3A6212E48204C1BE71A846E08DD2D9F1CBDDFF40CA00C10C62B1DD42486C70A09C454293BCA9ED4E7D6657E3F62076A14304943252A88EFA416770E0FBA270A141E7", "10001");

return crypter.RSAEncrypt(password);

}

}

## 迷你QQ邮箱登录算法

   QQ邮箱一直是群发人员的最爱,想当年多少人前赴后继研究去研究QQ邮箱登录.当然现在还有,这里我给大家介绍的,是从(迷你)微型邮箱登录.因为微型邮箱登录速度快,这对群发来说是一种不错的选择.很多人曾经付费求助这种协议,在本站就全部免费给大家讲解咯!http://cache.soso.com/img/img/e113.gif

http://w.mail.qq.com/cgi-bin/loginpage?f=xhtml

   登录自己的号码并抓包发现,他POST提交的地址如下:

http://w.mail.qq.com/cgi-bin/login?sid=

device =  
f =xhtml   
delegate\_url =  
action =  
tfcont =  
uin =397198519  
aliastype =@qq.com   
pwd =  
mss =1   
btlogin =登录   
ts =1381028909   
p =fSIPpM6LoWn2kCRiLRjlw449mi6717xltcXiohzduUE/n5H5Hxu9zdUVlzxYFMLrvhCMKgCopZJUe000PgfCoJ/kjqTcTlKcg+==

以上的参数除了ts和p我猜各位应该都知道怎么做,这个ts就是我们当前登录的时间戳.而P是什么?经过跟踪网页的js后发现他是

[JavaScript] 纯文本查看 复制代码

loginForm.onsubmit = function(){

var pwd = document.getElementById("pwd");

var p = document.getElementById("p");

var tsValue = document.getElementById("ts").value;

var PublicKey = "CF87D7B4C864F4842F1D337491A48FFF54B73A17300E8E42FA365420393AC0346AE55D8AFAD975DFA175FAF0106CBA81AF1DDE4ACEC284DAC6ED9A0D8FEB1CC070733C58213EFFED46529C54CEA06D774E3CC7E073346AEBD6C66FC973F299EB74738E400B22B1E7CDC54E71AED059D228DFEB5B29C530FF341502AE56DDCFE9";

var RSA = new RSAKey();

RSA.setPublic(PublicKey, "10001");

var Res = RSA.encrypt(pwd.value+ '\n' + tsValue + '\n');

if (Res)

{

p.value = hex2b64(Res);

pwd.value = "";

}

return true;

}

我想这里大家都应该已经很清楚加密方式了,  
他是先将 ("密码"+"\n"+ts的值+"\n")用公钥为publickey..的rsa加密算法进行加密.然后用加密结果用base64编码方式进行编码.  
rsa加密算法可以参考 [QQ注册涉及到的rsa加密算法[.net]](http://www.10qf.com/thread-5-1-1.html)  ,算法是一模一样的,只是公钥需要改成我们这里的publickey的值.  
这里是登录密码验证部分,接着,重头戏来了.登录返回的页面看不到登录成功的特征了吧?

<div class="status"> 正在登录，请稍候...</div>

我想这里正是很多人都卡住的地方,仔细看了一下返回的html代码,发现这是一个头部有重定向标记的代码.

<meta http-equiv="Refresh" content="0;url=http://w.mail.qq.com/cgi-bin/today?sid=nygWfblt408fITZkUDPT5-Nx,4,zn6ZTYD5w&first=1&mcookie=disabled"/>

对了,我们只要再用这个的地址请求一下就能很明显地看出是登录成功了还是没有.这就是邮箱的登录部分了.对邮件发送的更多细节有兴趣的朋友可以联系我.