def pkt\_split(pkt):

'''原生包拆分'''

pkt = repr\_hex(str(pkt)) # 对发送出的数据包16进制字符串转码

# 以下是对发送出的数据包拆分

Title = pkt[0:14],

#

Header = pkt[14:34]

#

Param = pkt[34:62]

#

Data = pkt[62:]

# 以下是对发送出的数据包进一步拆分 pkt\_split

ps = {

##Title

'Conn':Title[:],

## Header

'PID': Header[0:2], # Protocol ID

'ROSCTR': Header[2:4],

'RID': Header[4:8], # Redundancy Idetification

'PDUR': Header[8:12], # Potocol Data Unit referencr

'Para\_Lenth': Header[12:16],

'Data\_Lenth': Header[16:20],

## Param

'Func': Param[0:2],

'ItemCount': Param[2:4],

'Items': Param[4:],

## Data

'Data' : Data[:],

}

return ps

def filter(dataSet, pkt\_split, obj):

'''

筛包

$

obj={

'Conn',

'PID',

'ROSCTR',

'RID',

'PDUR',

'Para\_Lenth',

'Data\_Lenth',

'Func',

'ItemCount',

'Items',

'Data'}

'''

obj\_flag={'Conn':0, 'PID':1, 'ROSCTR':2, 'RID':3, 'PDUR':4, 'Para\_Lenth':5, 'Data\_Lenth':6, 'Func':7, 'ItemCount':8, 'Items':9, 'Data':10}

lenth = len(dataSet[0][obj\_flag[obj]])

for i in range(len(dataSet)):

compa = Pre.binCount(

Pre.str\_bw\_xor(

dataSet[i][obj\_flag[obj]], pkt\_split[obj])

) # 过筛

if compa < math.ceil(0.5\*lenth): # 不同的位不足总位数一半的视为不合格

return False

return True

# access = True

# loop\_max=0

# while access:

# loop\_max+=1

# READ\_SZL\_PACKET = TPKT() / COTPDT(EOT=1) / S7Header(ProtocolId=0x32, ROSCTR="Job", # 0x32 S7comm协议

# Parameters=S7WriteVarParameterReq( # S7写入参数请求包

# ItemCount=RandByte(), # 0x76,对应Items

# Items=S7WriteVarItemsReq(

# ItemCount=0x0003, # 对应Data

# BlockNum=0x0001)), # DB块号

# Data=S7WriteVarDataItemsReq(

# TransportSize=0x05, # 0x05写入

# Data='\x01\x02\x16' # 对应ItemCount，有3段

# )

# ) # 提交SNAP工作内容，发送出的数据包

# # TODO

# # READ\_SZL\_PACKET["S7Header"].show()

# rule = InitSet(params['FILTER']) # 过滤规则

# # dataSet = sc('./701\_selected.csv' .readCSV() # 读取701\_selected.csv数据集

# dataSet = Pre.sc(rule.seed).readCSV()

# ps = Pre.psp.pkt\_split(READ\_SZL\_PACKET)

# # print filter(dataSet,ps,'ItemCount')

# # compa = Pre.binCount(Pre.str\_bw\_xor(

# # dataSet[0][8], ps['ItemCount'])) # 过筛，异或运算筛选

# if Pre.pf.filter(dataSet,ps,params['FILTER\_OBJ']) or loop\_max > 20: # 超过一定筛选次数，不论是否合格，依旧发送

# print loop\_max

# access = False