快报 天天快报

2018年工业信息安全技能大赛解题报告-工业网络数据分析

北京网藤科技有限公司
2018-07-23 18:13

AppStore
Android
扫码下载快报

当前，随着中国制造2025战略深入推进，工业控制系统从单机走向互联、从封闭走向开放、从自动化走向智能化，安全漏洞和隐患不断涌现、安全事件频繁发生。我国面临的工业信息安全形势日益严峻。

▲ 图1 2018年工业信息安全技能大赛（东北赛区）开幕式

2018年工业信息安全技能大赛（东北赛区）分为漏洞挖掘赛和夺旗赛，其夺旗赛大概可分为**工业网络数据分析、工控固件逆向、工控安全分析**三个部分。

本文将介绍工业控制网络流量部分的解题思路，当然本文的思路不一定是最好的。仅仅作为抛砖，欢迎大家联系网藤科技“藤”安全实验室，索取赛题来玩！

0x01电力工控数据分析1（200分）

1赛题描述

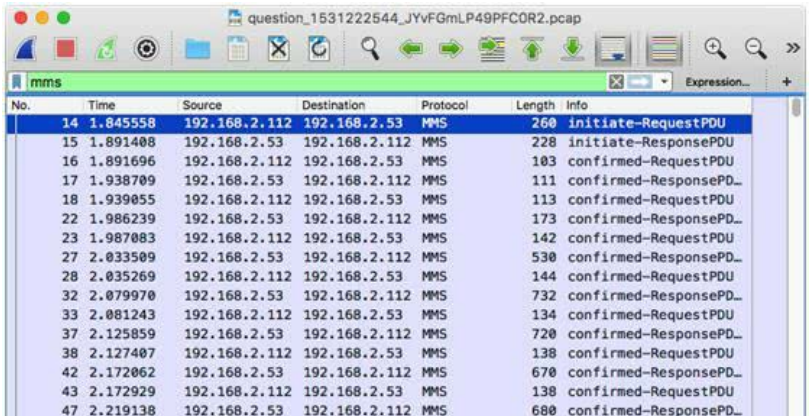
赛题如图2所示，附件：question_1531222544_JYvFGmLP49PFC0R2.pcap。

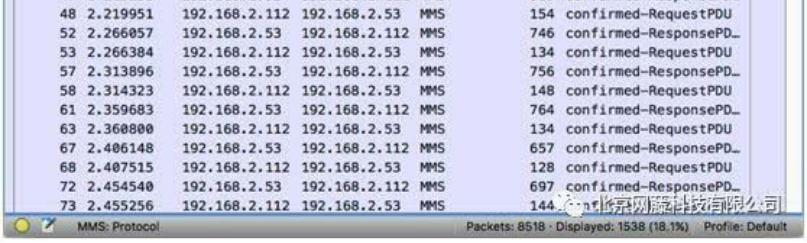
▲ 图2赛题

2解题思路

考虑到题目是了解MMS规约，发现数据中隐藏的Flag，所以可以先排除非MMS协议的数据包。

用wireshark打开数据包并选择MMS协议进行过滤，如图3所示：





▲ 图3过滤后MMS协议的数据包

经过MMS协议过滤后，共有1838个MMS协议的数据包，发现共有四个PDU（协议数据单元），分别是：Initiate-RequestPDU (启动-请求PDU)、Initiate-ResponsePDU (启动-应答PDU)、Confirmed-RequestPDU (确认-请求PDU)、Confirmed-ResponsePDU (确认-应答PDU)，所以把分析的重点放在Confirmed-RequestPDU和Confirmed-ResponsePDU中。

通过分析数据包得知MMS协议的规约中Confirmed-RequestPDU和Confirmed-ResponsePDU的结构，如下：

Confirmed-RequestPDU包含invokeID、listOfModifiers、confirmedServiceRequest、Request-detail四个部分组成；

Confirmed-ResponsePDU包含invokeID、confirmedServiceResponse、Response-detail三个部分组成。

接下来，先分析一下数据包中的MMS服务，编写脚本提取出数据包中所用的MMS服务并统计，脚本代码如下：

```
import pyshark

def get_service():
    try:
        captures = pyshark.FileCapture("question_1531222544_JYvFGmLP49PFC0R2.pcap")
        confirmed_services_request = {}
        confirmed_services_response = {}
        for capture in captures:
            for pkt in capture:
                if pkt.layer_name == "mms":
                    if hasattr(pkt, "confirmedservicerequest"):
                        service = pkt.confirmedservicerequest
                        if service in confirmed_services_request:
                            confirmed_services_request[service] += 1
                        else:
                            confirmed_services_request[service] = 1
                    if hasattr(pkt, "confirmedserviceresponse"):
                        service = pkt.confirmedserviceresponse
                        if service in confirmed_services_response:
                            confirmed_services_response[service] += 1
                        else:
                            confirmed_services_response[service] = 1
        # print
        print(confirmed_services_request)
        print(confirmed_services_response)
    except Exception as e:
```

脚本运行结果如下：

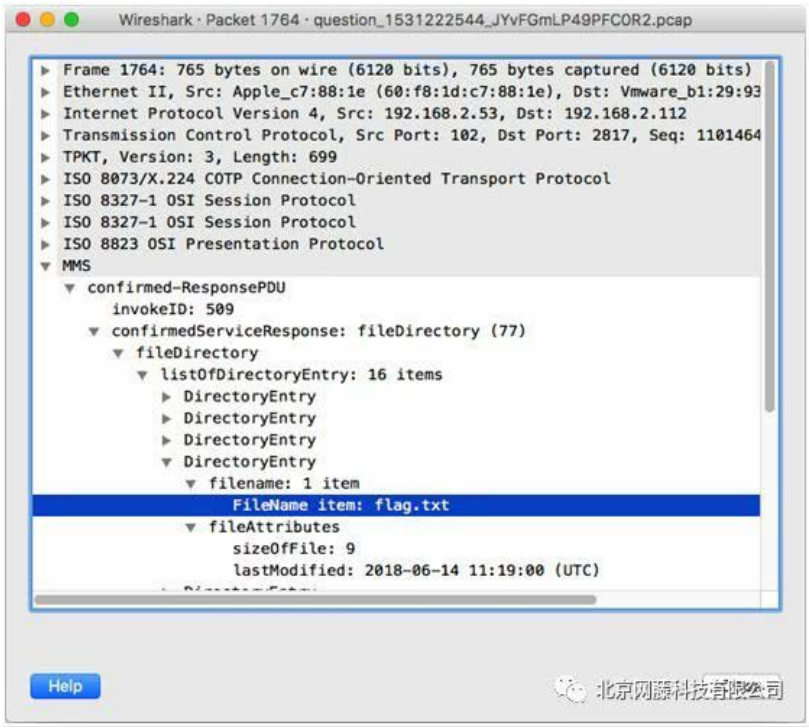
通过运行上面的脚本，发现数据包中使用到了9个服务，分别是

1 (getNameList)、4 (read)、5 (write)、6 (getVariableAccessAttributes)、12 (getNamedVariableListAttributes)、72 (fileOpen)、73 (fileRead)、74 (fileClose)、77 (fileDirectory)。

ConfirmedServiceRequest和ConfirmedServiceResponse的服务类型，

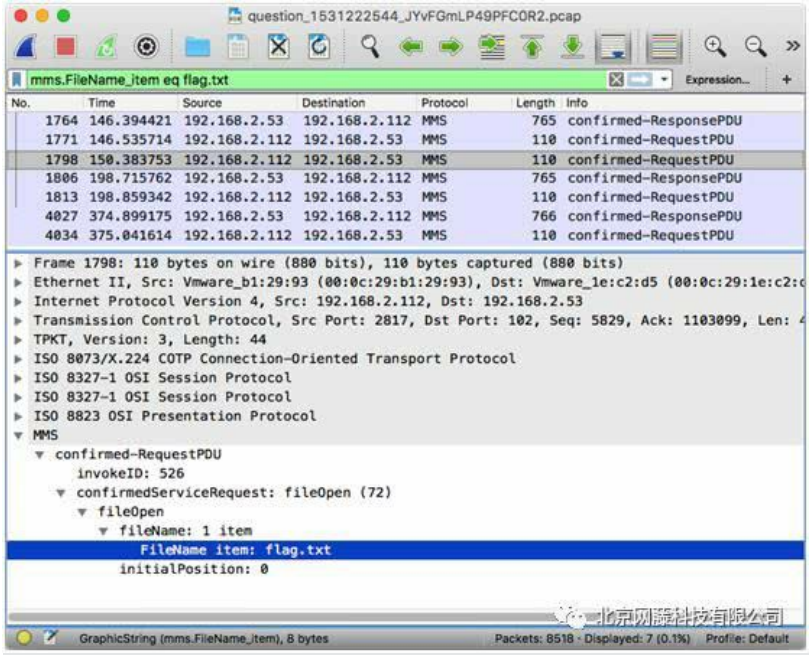
，可以参考GBT16720.2-2005工业自动化系统制造报文规范第2部分协议规范.pdf。

接下来就是对服务一个一个的分析，按照服务的出现次数从小到大来分析，先分析77 (fileDirectory)服务的数据包，惊奇的发现数据包第1764个比数据包第266个多一个flag.txt文件，而且文件最后一次修改时间为2018-06-14 11:19:00(UTC)，如图4所示：



▲ 图4获取文件目录的确认应答

可以猜测这个flag.txt文件就是我们所需要的flag信息，接着过滤出与flag.txt文件相关的数据包，如图5所示：



▲ 图5过滤后与flag.txt相关的数据包

从图5中发现数据包中存在读取flag.txt的请求，编写脚本找出flag.txt的文件内容，脚本代码如下：

```
import pyshark,
,
```

```
def flag():
    try:
        captures = pyshark.FileCapture("question_1531222544_JYv
FGmLP49PFC0R2.pcap")
        flag_frsm = False
        flag_frsm_id = None
        flag_read = False
        for capture in captures:
            for pkt in capture:
                if pkt.layer_name == "mms":
```

北京网藤科技有限公司

```
                    # file open
                    if hasattr(pkt, "confirmedservicerequest") an
d int(pkt.confirmedservicerequest) == 72:
                        if hasattr(pkt, "filename_item"):
                            filename_items = pkt.filename_item.fi
elds
                            for f in filename_items:
                                file_name = str(f.get_default_valu
e())
                                if file_name == "flag.txt":
                                    flag_frsm = True
                    if hasattr(pkt, "confirmedserviceresponse") a
nd int(pkt.confirmedserviceresponse) == 72 and flag_frsm:
                        # print(pkt.field_names)
                        if hasattr(pkt, "frsmid"):
                            flag_frsm_id = pkt.frsmid
                        flag_frsm = False
                    # file read
                    if hasattr(pkt, "confirmedservicerequest") an
d int(pkt.confirmedservicerequest) == 73 and flag_frsm_id:
                        if hasattr(pkt, "fileread"):
                            if str(pkt.fileread) == str(flag_frsm
_id):
                                flag_read = True
                        flag_frsm_id = None
                    if hasattr(pkt, "confirmedserviceresponse") a
nd int(pkt.confirmedserviceresponse) == 73 and flag_read:
                        if hasattr(pkt, "filedata"):
                            data = str(pkt.filedata).replace(":",
"")
                            print(hex_to_ascii(data))
                        flag_read = False
```

北京网藤科技有限公司

```
    except Exception as e:
        print(e)


def hex_to_ascii(data):
    data = data.decode("hex")
    flags = []
    for d in data:
        _ord = ord(d)
        if (_ord > 0) and (_ord < 128):
            flags.append(chr(_ord))
```

```
    return ''.join(flags)

if __name__ == '__main__':
    flag()
```

北京网藤科技有限公司

脚本运行结果如下：

61850@102

所以，61850@102就是需要寻找的Flag

0x02工业协议数据分析（300分）

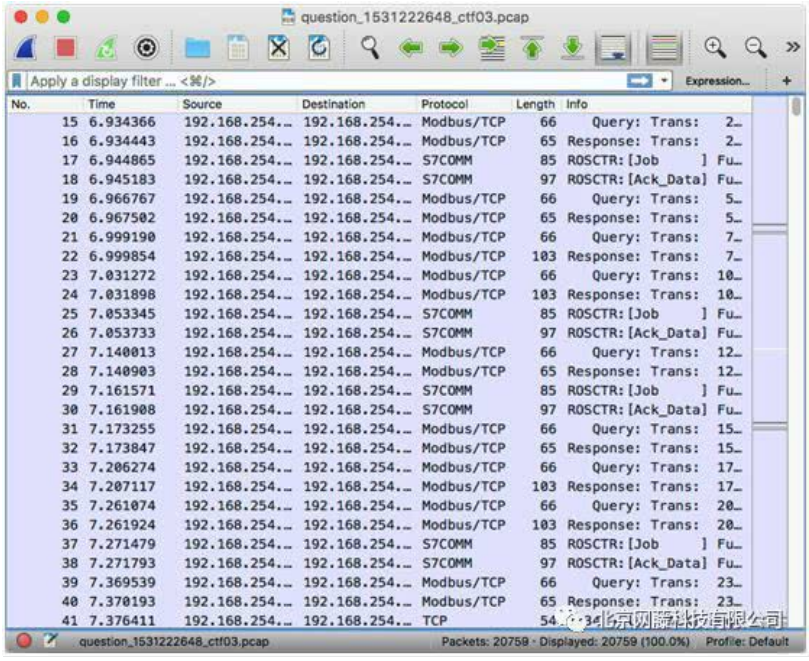
1赛题描述

▲ 图6赛题

2解题思路

用wireshark打开下载到的数据包后可以发现数据包中有大量的数据混杂在一起，考虑到题目是工业协议数据分析，因此可以初步排除非工控的协议，留到最后分析。

经过逐步排除后可以发现，数据包中相关的工控流量有Modbus/TCP协议（端口是TCP 502）和西门子S7comm协议（端口是TCP 102），如图7所示：



北京网藤科技有限公司

▲ 图7数据包中的工控流量

接下来只需要进一步分析Modbus/TCP协议及西门子S7comm协议的数据包即可。

Ok，先来分析西门子S7comm协议吧，从数据包中可以发现，西门子S7comm协议中的PDU类型有Job和Ack_Data两种，所以分析起来更加简单了，编写脚本解析出数据包中使用到的西门子S7comm协议的功能码，脚本代码如下：

```
import pyshark

def get_func_s7():
    try:
```

```
        captures = pyshark.FileCapture("question_1531222648_ctf
03.pcap")
        func_codes = {}
        for c in captures:
            for pkt in c:
                if pkt.layer_name == "s7comm":
                    if hasattr(pkt, "param_func"):
                        func_code = pkt.param_func
                        if func_code in func_codes:
                            func_codes[func_code] += 1
                        else:
                            func_codes[func_code] = 1
        print(func_codes)
    except Exception as e:
        print(e)

if __name__ == '__main__':
    get_func_s7()
```

北京网藤科技有限公司

脚本执行后，数据包中所使用的西门子S7comm协议的功能码有0xf0 (Setupcommunication)（出现2次）和0x04 (Read Var)（出现6196次），但是进一步分析后，发现大量的0x04 (Read Var)都是对DB 1.DBX 4.0进行的读取行为，而且数据（数据是0000000020037000000000c000000002c41）没有变化。因此也可以大致排除西门子S7comm协议中存在Flag的可能。

下一步需要分析的就是Modbus/TCP协议了，同样编写脚本解析出数据包中使用到的Modbus/TCP协议的功能码，脚本代码如下：

```
import pyshark

def get_func_mb():
    try:
        captures = pyshark.FileCapture("question_1531222648_ctf
03.pcap")
        func_codes = {}
        for c in captures:
            for pkt in c:
                if pkt.layer_name == "modbus":
                    func_code = int(pkt.func_code)
                    if func_code in func_codes:
                        func_codes[func_code] += 1
                    else:
                        func_codes[func_code] = 1
        print(func_codes)
    except Exception as e:
        print(e)

if __name__ == '__main__':
    get_func_mb()
```

北京网藤科技有限公司

脚本执行后，数据包中所使用的Modbus/TCP协议的功能码有1 (Read coils)（出现3220次）、2 (Read Discrete Inputs)（出现3220次）、3 (Read HoldingRegisters)（出现3220次）、4 (Read Input Registers)（出现3220次）和16 (Write Multiple Registers)（

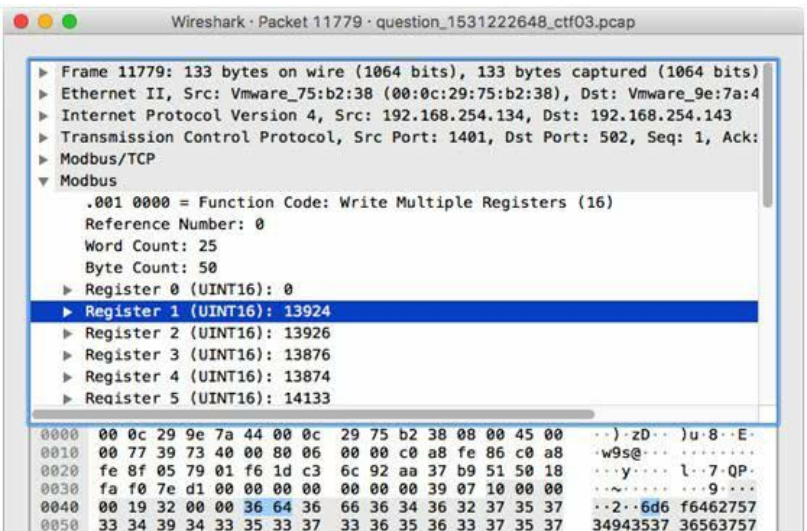
出现228次)。而功能码中的1 (Read coils)、2 (Read DiscreteInputs)、3 (Read Holding Registers)、4 (Read InputRegisters)的请求都出现过3220次，且请求的各个寄存器数值一直在不停的变化，看起来比较像正常的工业数据。

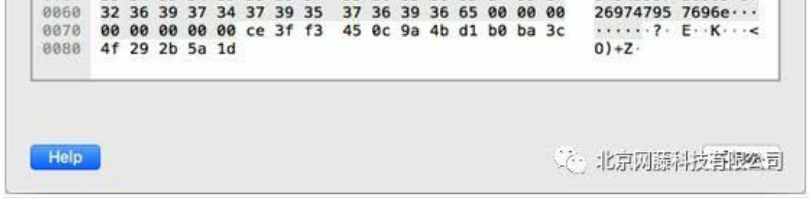
所以，接下来编写脚本对16 (WriteMultiple Registers)功能码相关的数据包进行分析，脚本代码如下：

```
import pyshark

def find_flag():
    try:
        cap = pyshark.FileCapture("question_1531222648_ctf03.pcap")
        idx = 1
        for c in cap:
            for pkt in c:
                if pkt.layer_name == "modbus":
                    func_code = int(pkt.func_code)
                    if func_code == 16:
                        payload = str(c["TCP"].payload).replace(":", "")
                        print(hex_to_ascii(payload))
                        print("{0} ##############################################".format(idx))
            idx += 1
    except Exception as e:
        print(e)

def hex_to_ascii(payload):
    data = payload.decode("hex")
    flags = []
    for d in data:
        _ord = ord(d)
        if (_ord > 0) and (_ord < 128):
            flags.append(chr(_ord))
    return ''.join(flags)

if __name__ == '__main__':
    find_flag()
```

北京网藤科技有限公司

脚本执行后，发现第11779个数据包比较异常，先看一下它的TCP Payload是926d6f6462757349435373656375726974795769 6e?EK(如图8所示)：





▲ 图8隐藏Flag的数据包

可以大胆的猜测6d6f6462757349435373656375726974795769 7e就是我们需要的Flag，将其16进制字符串转成ASCII字符串后得到modbusICSsecurityWin，所以modbusICSsecurityWin就是需要寻找的Flag

0x03 Modbus协议分析1（200分）

1赛题描述

赛题如图9所示，附件：question_1531222756_modbus1.pcap。

▲ 图9赛题

2解题思路

考虑到题目是Modbus协议数据分析，因此可以初步排除非Modbus的协议。

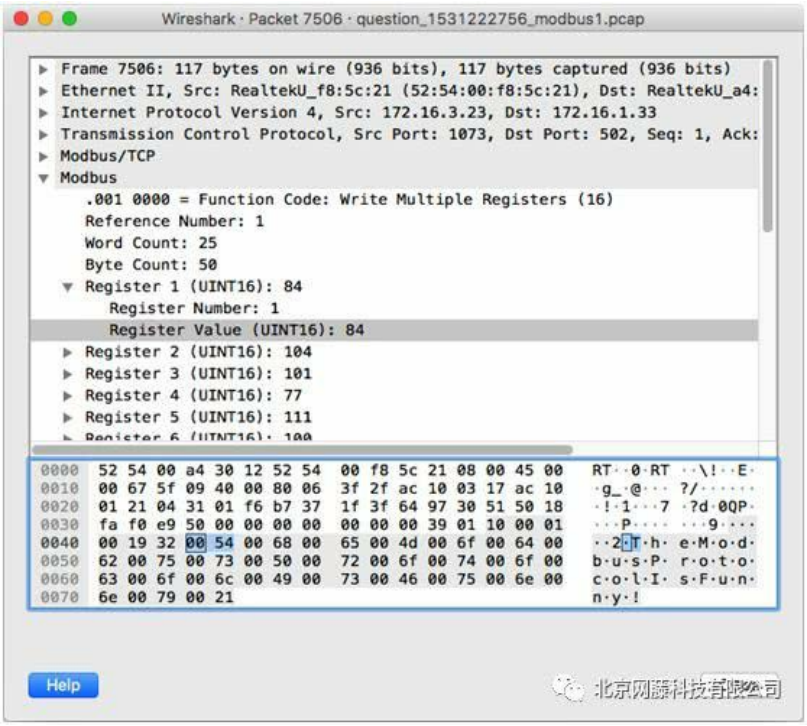
先分析一下数据包中所有使用到的Modbus/TCP协议功能码，同样编写脚本对其进行分析，脚本代码如下：

```
import pyshark

def get_func_mb():
    try:
        captures = pyshark.FileCapture("question_1531222756_modbus1.pcap")
        func_codes = {}
        for c in captures:
            for pkt in c:
                if pkt.layer_name == "modbus":
                    func_code = pkt.func_code
                    if func_code in func_codes:
                        func_codes[func_code] += 1
                    else:
                        func_codes[func_code] = 1
        print(func_codes)
    except Exception as e:
```


脚本执行后，数据包中所使用的Modbus/TCP协议的功能码有1 (Read coils)（出现702次）、2 (Read Discrete Inputs)（出现702次）、3 (Read HoldingRegisters)（出现702次）、4 (Read Input Registers)（出现702次）和16 (Write Multiple Registers)（出现2次）。而功能码中的1 (Read coils)、2 (Read DiscreteInputs)、3 (Read Holding Registers)、4 (Read InputRegisters)的请求都出现过702次，且请求的各个寄存器数值一直在不停的变化，看起来比较像正常的工业数据。

而16 (WriteMultiple Registers)请求只出现了2次，因此比较可疑。数据包中的16 (WriteMultiple Registers)写入到25个连续的寄存器中（如图10所示），我们所需要的fla

g信息就可能藏在这里。



▲ 图10隐藏Flag的数据包

所以，TheModbusProtocolIsFunny!就是需要寻找的Flag

0x04 S7comm协议分析1（200分）

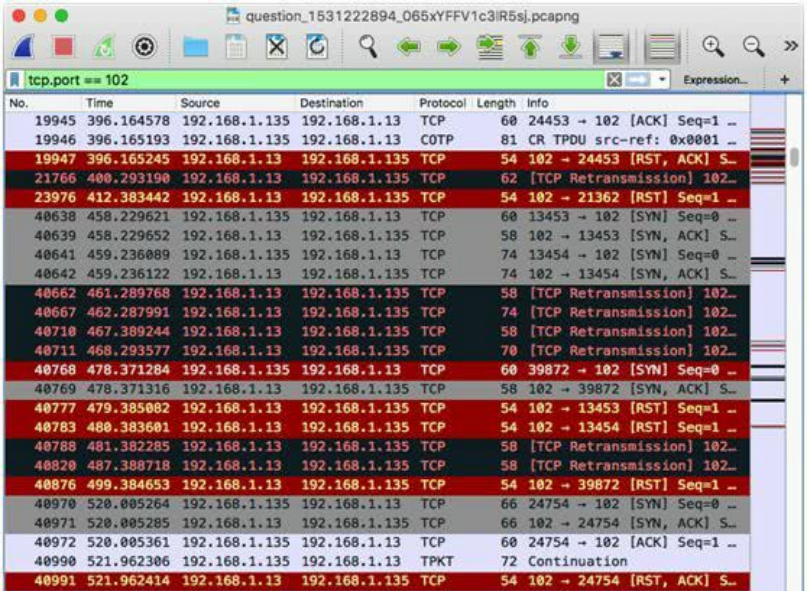
1赛题描述

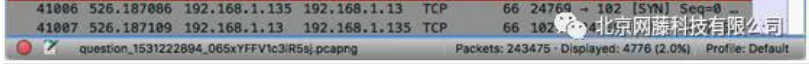
▲ 图11赛题

2解题思路

考虑到题目是S7comm协议数据分析，因此可以初步排除非S7comm的协议。

如下图12所示，可以看到该流量中存在大量的重传现象，因此一种可能是网络中存在中间人攻击，传输的流量可能被截取篡改，而篡改的数据包中就很有可能存在需要的Flag。





▲ 图12数据重传现象

此时可以将过滤后的数据包使用wireshark导出(s7.pcapng)后用scapy工具进行辅助分析是否存在中间人改包及定位数据包位置。

```
import scapy.all as scapy
from scapy.layers.inet import TCP

if __name__ == '__main__':
    try:
        captures = scapy.rdpcap("s7.pcapng")
        for i in range(len(captures) - 1):
            pkt1 = captures[i]
            pkt2 = captures[i + 1]
            # 只检测 push 的 TCP 数据包
            if pkt1[TCP].flags == 0x18 and pkt2[TCP].flags == 0x
18:
```

北京网藤科技有限公司

```
                # 判断 ack 和 seq 重复的数据包
                if pkt1[TCP].seq == pkt2[TCP].seq and pkt1[TCP].
ack == pkt2[TCP].ack:
                    if pkt1.load != pkt2.load:
                        print("Find target: packet number: %s" %
i + 1)
    except Exception as e:
        print(e)
```

北京网藤科技有限公司

通过辅助分析后，并没有在数据包中发现被篡改的数据，因此可以排除中间人改包的这个可能。怀疑造成数据包显示大量重放的原因可能与端口镜像配置有关。

接下来就是进一步分析S7comm协议的数据包，但是数据包中没有S7comm协议相关的数据包，有的是COTP协议相关的数据包，所以接下来就是分析COTP协议，先统计一下cotp协议的PDU类型，编写脚本如下：

```
import pyshark

if __name__ == '__main__':
    try:
        captures = pyshark.FileCapture("s7.pcapng")
        pdu_types = {}
        for c in captures:
            for pkt in c:
                if pkt.layer_name == "cotp":
                    if hasattr(pkt, "type"):
                        type = pkt.type
                        if type in pdu_types:
                            pdu_types[type] += 1
                        else:
                            pdu_types[type] = 1
        print(pdu_types)
    except Exception as e:
```

北京网藤科技有限公司

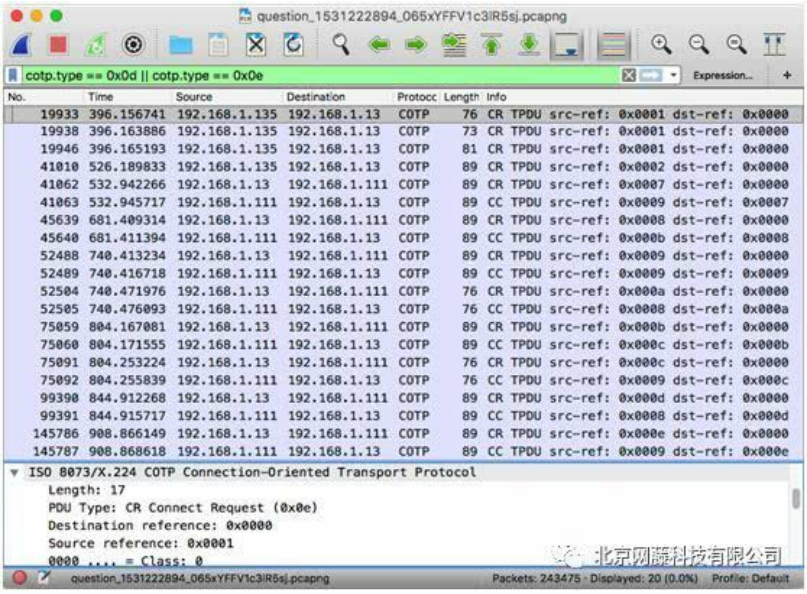
脚本执行后，数据包中所使用的COTP协议的PDU类型有0x0e (CR Connect Request, 连接请求)(出现12次)、0x0d (CC Connect Confirm, 连接确认)(出现8次)和0x0f (DT Data, 数据传输)(出现3696次)。从统计的结果来看，一共有12次连接请求，但

是只有8次连接确认，所以推测有的连接存在非法信息，这可能就是所需要的Flag。

COTP（ISO 8073/X.224 COTP Connection-Oriented Transport Protocol）是OSI 7层协议定义的位于TCP之上的协议。COTP以“Packet”为基本单位来传输数据，这样接收方会得到与发送方具有相同边界的数据。

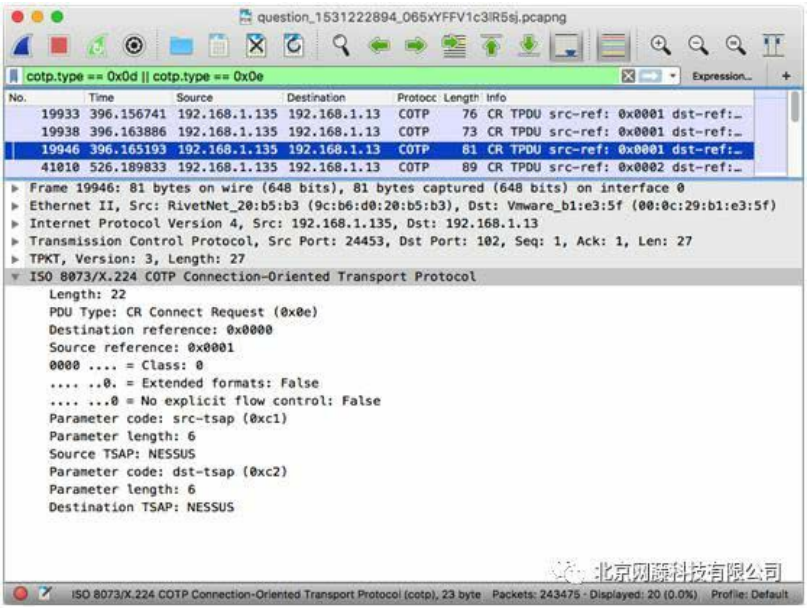
COTP协议分为两种形态，分别是COTP连接包（COTP Connection Packet）和COTP功能包（COTP Fuction Packet）。

用wireshark过滤后，如图13所示:



▲ 图13 PDU类型是0x0e和0x0d的相关数据包

发现第19933、19938、19946、41010个数据包只进行了连接请求，没有得到192.168.1.13的连接确认，而且19933、19938、19946的src-ref都是0x0001，先重点分析这3个数据包，果然在第19946个数据包中发现了猫腻，如图14所示:



▲ 图14隐藏Flag的数据包

通过对比分析其他16个数据包，可以确定NESSUS就是需要的Flag

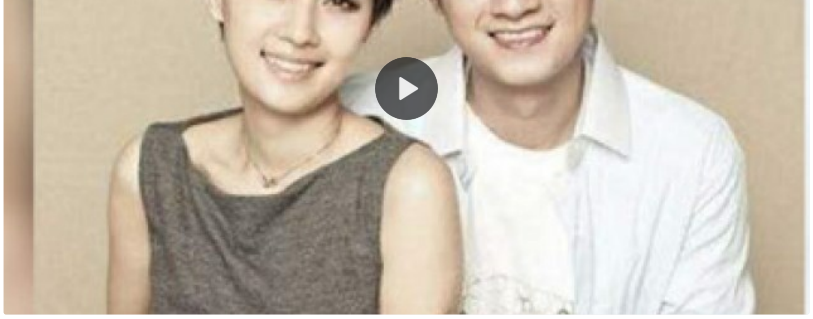
0x05参考

GBT16720.2-2005工业自动化系统制造报文规范第2部分协议规范.pdf

CTF WP –工控业务流量分析-工匠安全实验室

热点视频

一别两宽 各生欢喜：2分钟回顾文章马伊琍14年相知路



精选 热文

五年前就能退休，单位为何不通知我？十几万的损失找谁要？



湖北综合　122评论

四川电视台道歉，“暗访”按摩店节目出现不当画面

湖北日报

越南四千亿高铁大单选择日本，我国落选，如今越南的近况让人唏嘘


两碗茶　21评论

当了李嘉诚儿子11年保镖，张子强被捉之后，他却被无情炒了鱿鱼



lyn的学习日记　1212评论

叙叛军司令被俄战机炸死，屏蔽信号成关键，美见识到俄军厉害



军情堡

张扣扣被执行死刑，王家的赔偿到位了么？

谭谭文化　867评论

9月1日起，摩托车迎来好消息，老百姓：终于可以踏实出门了！

思思聊说汽车