

一道MMS工控协议CTF题的WriteUp

原创

p4ndat 于 2019-07-06 09:42:09 发布 3495 收藏 11

分类专栏: [CTF](#) 文章标签: [工控安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/cn_lyxc/article/details/94832631

版权



[CTF 专栏收录该内容](#)

1 篇文章 0 订阅

订阅专栏

0x01 赛题说明

赛题说明

[附件下载](#)

智能变电站通过61850规约进行监控层到间隔层的数据采集, 请分析网络数据包, 了解MMS规约, 进一步发现数据中隐藏的flag。

赛题说明:

只能变电站通过 61850 规约进行监控层到间隔层的数据采集, 请分析网络数据包, 了解 MMS 规约, 进一步发现数据中隐藏的 flag。

题目:question_1531222544_JYvFGmLP49PFC0R2.pcap.zip(下载见最下方的百度云传送门)

0x02 背景知识

MMS即制造报文规范, 是ISO/IEC9506标准所定义的一套用于工业控制系统的通信协议。

此处不介绍关于 MMS 协议的具体内容, 只做报文简单讲解。

如果有兴趣, 可以阅读此文: MMS (见最下方的百度云传送门)

1、Initiate

```
MMSpdu Received ::=
A8 25 80 02 08 00 81 01 05 82 01 05 83 01 05 A4 16 80 01 01 81 03 05 F8 00 82 0C 03 EE 19 00 18 00 02 00 00 00 F
D 18

Tag Length Value -- Tag definition
-----
```

```

[8] A8 25 -- initiate-RequestPDU,

A8 = 1010 1000;

bit7,6 = Tag type, 00 = Universal tag, 10 = Context specific

bit5 = , 0 = Primitive, 1 = Constructed

(
    00 0 = INTEGER, BITSTRING, BOOLEAN

    00 1 = SEQUENCE, SEQUENCE OF

    10 0 = IMPLICIT

    10 1 = IMPLICIT SEQUENCE, IMPLICIT SEQUENCE OF
)

bit4-0 = Value, for primitive Universal tags, value defined in ASN.1, other use [x] in MMS.

{

    [0] 80 02 08 00 - localDetailCalling (maxProposedMMSPduSize) = 2048 bytes

    [1] 81 01 05 -- proposedMaxServOutstandingCalling

    [2] 82 01 05 -- proposedMaxServOutstandingCalled

    [3] 83 01 05 -- proposedDataStructureNestingLevel

    [4] A4 16 -- mmsInitRequestDetail

    {

        [0] 80 01 01 -- proposedVersionNumber, MMS ISO IS 9506

        [1] 81 03 05 F8 00 -- proposedParameterCBB,

        BitString(11 bits used)

        05 = indicate number of unused bit

        F8 00

        {

            str1 (bit 0 / array support / MSB of F8) -- supported

            str2 (bit 1 / structure support) -- supported

            vnam (bit 2 / named variable support) -- supported

            valt(bit 3 /alternate access support) -- supported
        }
    }
}

```

```
vadr(bit 4/ unnamed variable support) -- supported

viscera(bit 5/ scattered access support) -- not-supported

toy(bit 6/ third party operations support) -- not-supported

villas(bit 7/ named variable list support) -- not-supported

real(bit 8 / ASN.1 real data type support) -- not-supported

ache(bit 9/ acknowledge event conditionsupport) -- not-supported

chi(bit 10 / condition event support) -- not-supported

}

[2] 82 0C 03 EE 19 00 18 00 02 00 00 00 FD 18 -- servicesSupportedCalling, see ISO/IEC-9506

}

}
```

2、Initiate-Response

```
MMSpdu Received ::=

A9 25 80 02 08 00 81 01 05 82 01 05 83 01 05 A416 80 01 01 81 03 05 F8 00 82 0C 03 EE 19 00 18 00 02 00 00 00
FD 18

[9] A9 25 -- Initiate-ResponsePDU

{

  [0] 80 02 08 00 -- localDetailCalled

  [1] 81 01 05 -- negotiatedMaxServOutstandingCalling

  [2] 82 01 05 -- negotiatedMaxServOutstandingCalled

  [3] 83 01 05 -- negotiatedDataStructureNestingLevel

  [4] A4 16

  {

    [0] 80 01 01 -- negotiatedVersionNumber

    [1] 81 03 05 F8 00 -- negotiatedParameterCBB

    [2] 82 0C 03 EE 19 00 18 00 02 00 00 00 FD 18 -- servicesSupportedCalled

  }

}
```

3、Identify

MMSPdu Received ::=

A0 05 02 01 01 82 00

[0] A0 05 -- Confirmed-RequestPDU

```
{
    02 01 01 -- invokeID
    ( 02 = 000 0 0010, Universal Primitive tag, Tag Value = 2 = Integer. )
    [2] 82 00 -- ConfirmedServiceRequest, 82 = Request identify
}
```

where, invokeID ::=01

4. Identify-Response

MMSPdu Received ::=

A1 2A 02 01 01 A2 25 80 0B 53 49 53 43 4F 2C 2049 6E 63 2E 81 10 41 58 53 34 2D 4D 4D 53 2D 3133 32 2D 30 31 38 82 04 32 2E 30 30

[1] A1 2A -- Confirmed-ResponsePDU

```
{
    02 01 01 -- invokeID, Integer
    [2] A2 25 -- ConfirmedServiceResponse, A2 = Response identify.
    {
        [0] 80 0B 53 49 53 43 4F 2C 20 49 6E 63 2E -- vendorName
        [1] 81 10 41 58 53 34 2D 4D 4D 53 2D 31 33 32 2D 30 31 38 -- modelName
        [2] 82 04 32 2E 30 30 -- revision
    }
}
```

where,

invokeID::=01 note: matching of response to is done by matching invokeID with response invokeID.

vendorName::="SISCO, Inc"

modelName::="AXS4-MMS-132-018"

revision::="2.00"

check [MMS and ASN.1 Encoding] page 10.

5. Read-Request

```

MMSPdu Received ::=

A0 1E 02 01 0A A4 19 A1 17 A0 15 30 13 A0 11 80 0F 66 65 65 64 65 72 31 5F 33 5F 70 68 61 73 65

[0] A0 1E -- ConfirmedRequestPDU
{
    02 01 0A -- invokeID

    [4] A4 19 -- ConfirmedServiceRequest, A4 = Read
    {
        [1] A1 17 -- variableAccessSpecification
        {
            [0] 30 13 -- listOfVariable
            (30 = 00 1 10000, Universal Constructed)
            {
                [0] A0 11 -- variableSpecification
                {
                    [0] 80 0F 66 65 65 64 65 72 31 5f 33 5f 70 68 61 73 65 -- name
                }
            }
        }
    }
}

where,

invokeID ::= 0A

Identifier (name of variable to read) ::= "feeder1_3_phase"

```

6、Read -Response

```

Assume

typedef struct var_def
{
    int a;

    int b;
}

```

```
} VAR_DEF;
```

```
VAR_DEF feeder1_3_phase;
```

MMS Data Production

```
Data ::= CHOICE
```

```
{
```

```
    [1] IMPLICIT SEQUENCE OF,-- arrayed data
```

```
    [2] IMPLICIT SEQUENCE OF,-- structured data
```

```
    [3] IMPLICIT BOOLEAN,
```

```
    [4] IMPLICIT BIT STRING,
```

```
    [5] IMPLICIT INTEGER,-- signed int
```

```
    [6] IMPLICIT INTEGER,-- unsigned int
```

```
    [7] IMPLICIT Floating Point,
```

```
    [9] IMPLICIT OCTET STRING,
```

```
    [10] IMPLICIT VisibleString,
```

```
    [11] IMPLICIT GeneralizedTime,
```

```
    [12] IMPLICIT TimeOfDay,
```

```
    [13] IMPLICIT INTEGER,-- BCD
```

```
    [14] IMPLICIT BIT STRING,-- boolean array
```

```
    [15] IMPLICIT OBJECT IDENTIFIER
```

```
}
```

The encoded structure of the encoded data can be determined via VAR_DEF

```
VAR_DEF ::=          TAG
```

```
-----
```

```
struct {           A2
```

```
    inta;           85
```

```
    intb;           85
```

```
}
```

```

MMSPdu Received ::=
A1 0F 02 01 0A A4 0A A1 08 A2 06 85 01 00 85 0100

[1] A1 0F -- ConfirmedResponsePDU
{
    02 01 0A -- invokeID

    [4] A4 0A -- ConfirmedServiceResponse, A4 = Read
    {
        [1] OF A1 08 -- listOfAccessResult
        {
            A2 06 -- success, Data of struct
            {
                85 01 00 -- int a;
                85 01 00 -- int b;
            }
        }
    }
}

where,
invokeID ::= 0A
value of a ::= 00, value of b ::= 00

```

这里例举了 3 种 MMS 报文。

通过分析这三种报文得知 MMS 协议的规约中的一些结构，如下：

Initiate

initiate-RequestPD结构包含：

- localDetailCalling
- proposedMaxServOutstandingCalling
- proposedMaxServOutstandingCalled
- proposedDataStructureNestingLevel
- mmsInitRequestDetail
- proposedVersionNumber
- proposedParameterCBB
- servicesSupportedCalling 等信息

Initiate-ResponsePDU结构包含：

- localDetailCalled
- negotiatedMaxServOutstandingCalling
- negotiatedMaxServOutstandingCalled
- negotiatedDataStructureNestingLevel
- negotiatedVersionNumber
- negotiatedParameterCBB
- servicesSupportedCalled等信息

Identify:

Confirmed-RequestPDU结构包含:

- invokeID
- tag
- ConfirmedServiceRequest等信息

Confirmed-ResponsePDU结构包含:

- invokeID
- ConfirmedServiceResponse
- vendorName
- modelName
- revision等信息

Read:

ConfirmedRequestPDU结构包含:

- invokeID
- ConfirmedServiceRequest
- variableAccessSpecification
- listOfVariable
- variableSpecification
- name等信息

ConfirmedResponsePDU结构包含:

- invokeID
- ConfirmedServiceResponse
- listOfAccessResult等信息

这里仅作简单说明，实际上以上的三种报文并非包含所有的MMS协议的协议数据单元，也没有包含所有的结构

如果有新兴趣研究具体内容，可以下载阅读此文件：[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](#)（下载见最下方的百度云传送门）

0x03 解题

对于数据包分析的 CTF 题目，第一件做的事情就是搜索关键字：flag

The screenshot shows the Wireshark interface with a search filter 'flag' applied. The packet list pane shows several MMS packets. The packet details pane is expanded to show the 'FileName item: flag.txt' field. The packet bytes pane shows the raw data of the selected packet, with a red box highlighting the hex value '6062628Z 0' and its ASCII representation 'fl'.

No.	Time	Source	Destination	Protocol	Length	Info
1749	55.0661	192.168.2.53	192.168.2.112	COTP	1094	DT TPDU (0) [COTP fragment, 1021 bytes]
1750	55.0661	192.168.2.53	192.168.2.112	COTP	1094	DT TPDU (0) [COTP fragment, 1021 bytes]
1751	55.0661	192.168.2.112	192.168.2.53	TCP	66	2817 → 102 [ACK] Seq=5004 Ack=1098095 Win=238456 Len=0 TSval=745627 TSecr=638677
1752	55.0664	192.168.2.53	192.168.2.112	COTP	1094	DT TPDU (0) [COTP fragment, 1021 bytes]
1753	55.0664	192.168.2.53	192.168.2.112	COTP	1094	DT TPDU (0) [COTP fragment, 1021 bytes]
1754	55.0665	192.168.2.112	192.168.2.53	TCP	66	2817 → 102 [ACK] Seq=5004 Ack=1100151 Win=236400 Len=0 TSval=745627 TSecr=638677
1755	55.0666	192.168.2.53	192.168.2.112	COTP	1094	DT TPDU (0) [COTP fragment, 1021 bytes]
1756	55.0667	192.168.2.53	192.168.2.112	MMS	322	507 confirmed-ResponsePDU
1757	55.0667	192.168.2.112	192.168.2.53	TCP	66	2817 → 102 [ACK] Seq=5004 Ack=1101435 Win=235116 Len=0 TSval=745627 TSecr=638677
1758	55.0698	192.168.2.112	192.168.2.53	TCP	66	[TCP Window Update] 2817 → 102 [ACK] Seq=5004 Ack=1101435 Win=245396 Len=0 TSval=745627 TSecr=638677
1759	55.0708	192.168.2.112	192.168.2.53	TCP	66	[TCP Window Update] 2817 → 102 [ACK] Seq=5004 Ack=1101435 Win=256960 Len=0 TSval=745628 TSecr=638677
1760	55.0761	192.168.2.112	192.168.2.53	MMS	99	508 confirmed-RequestPDU
1761	55.1022	192.168.2.112	192.168.2.53	MMS	95	508 confirmed-ResponsePDU
1762	55.2904	192.168.2.112	192.168.2.53	TCP	66	2817 → 102 [ACK] Seq=5037 Ack=1101464 Win=256928 Len=0 TSval=745630 TSecr=638680
1763	146.353	192.168.2.112	192.168.2.53	MMS	102	509 confirmed-RequestPDU
1764	146.394	192.168.2.53	192.168.2.112	MMS	765	509 confirmed-ResponsePDU

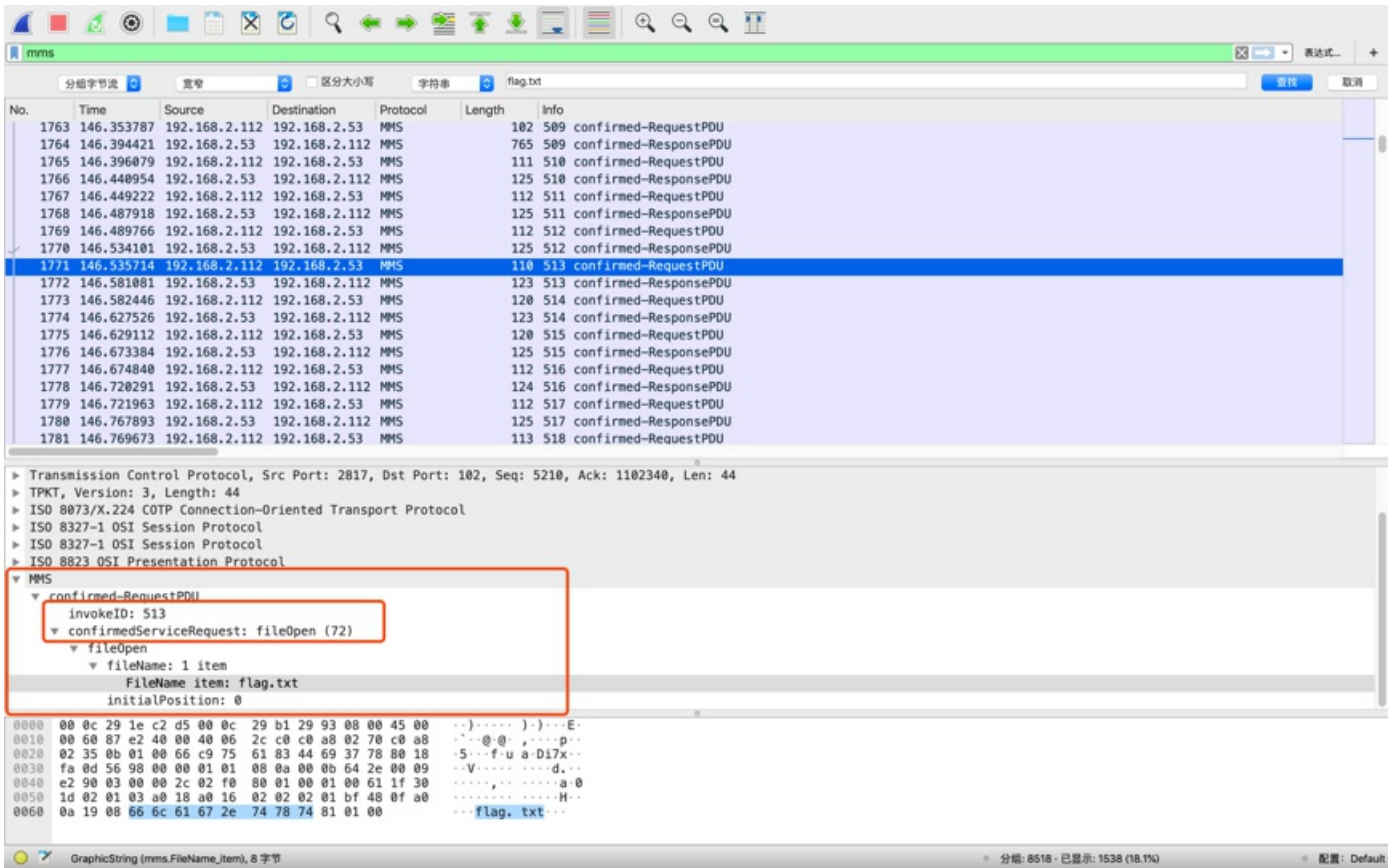
```
DirectoryEntry
DirectoryEntry
DirectoryEntry
DirectoryEntry
  filename: 1 item
    FileName item: flag.txt
      fileAttributes
      DirectoryEntry
      DirectoryEntry
      DirectoryEntry
      DirectoryEntry
      DirectoryEntry
```

Offset	Hex	ASCII
00e0	36 30 36 32 36 32 38 5a	0062628Z 0'
00f0	61 67 2e 74 78 74 a1 14	fl.txt.....201
0100	38 30 36 31 34 31 31 31	88614111 90070...

通过关键字的搜索可以发现存在 flag.txt，进一步查询此关键字：
在

```
1771 146.535714 192.168.2.112 192.168.2.53 MMS 110 513 confirmed-RequestPDU
```

发现：



如上图所示，这是一个RequestPDU，请求的内容如下：

```

MMS
  confirmed-RequestPDU
    invokeID: 513
    confirmedServiceRequest: fileOpen (72)
      fileOpen
        fileName: 1 item
          FileName item: flag.txt
        initialPosition: 0
  
```

对应的是RequestPDU结构

那么根据上文中的知识，存在着对应的 ResponsePDU

但是注意，我们需要的并不是fileOpen（72）的数据流

fileOpen的数据告诉我们的仅仅是打开文件的名字

这些文件中存在一个文件名叫 flag.txt的文件，而我们需要的是 fileRead 或是 fileWrite

根据上图中，flag.txt所在行为 1771，那么我们只需要用筛选器筛选出 fileRead 或者 fileWrite 的ResponsePDU

就应该在 1771 后 N 行找到 flag

根据 fileOpen 为 72 来看，73 应该为 fileRead，74 应该为 fileWrite，75 应该为 fileClose

但是发现

mms.confirmedServiceRequest == 73

分组字节流 宽窄 区分大小写 字符串 flag.txt

No.	Time	Source	Destination	Protocol	Length	Info
300	54.543475	192.168.2.112	192.168.2.53	MMS	99	498 confirmed-Req
451	54.618571	192.168.2.112	192.168.2.53	MMS	99	499 confirmed-Req
602	54.662412	192.168.2.112	192.168.2.53	MMS	99	500 confirmed-Req
754	54.707352	192.168.2.112	192.168.2.53	MMS	99	501 confirmed-Req
905	54.768299	192.168.2.112	192.168.2.53	MMS	99	502 confirmed-Req
1055	54.812376	192.168.2.112	192.168.2.53	MMS	99	503 confirmed-Req
1208	54.866546	192.168.2.112	192.168.2.53	MMS	99	504 confirmed-Req
1358	54.906909	192.168.2.112	192.168.2.53	MMS	99	505 confirmed-Req
1510	54.958874	192.168.2.112	192.168.2.53	MMS	99	506 confirmed-Req
1660	55.019700	192.168.2.112	192.168.2.53	MMS	99	507 confirmed-Req
1800	150.419434	192.168.2.112	192.168.2.53	MMS	99	527 confirmed-Req
4063	380.143719	192.168.2.112	192.168.2.53	MMS	99	1148 confirmed-Re
4216	380.222227	192.168.2.112	192.168.2.53	MMS	99	1149 confirmed-Re
4369	380.259169	192.168.2.112	192.168.2.53	MMS	99	1150 confirmed-Re
4422	382.295733	192.168.2.112	192.168.2.53	MMS	99	1153 confirmed-Re
4517	382.949830	192.168.2.112	192.168.2.53	MMS	99	1156 confirmed-Re

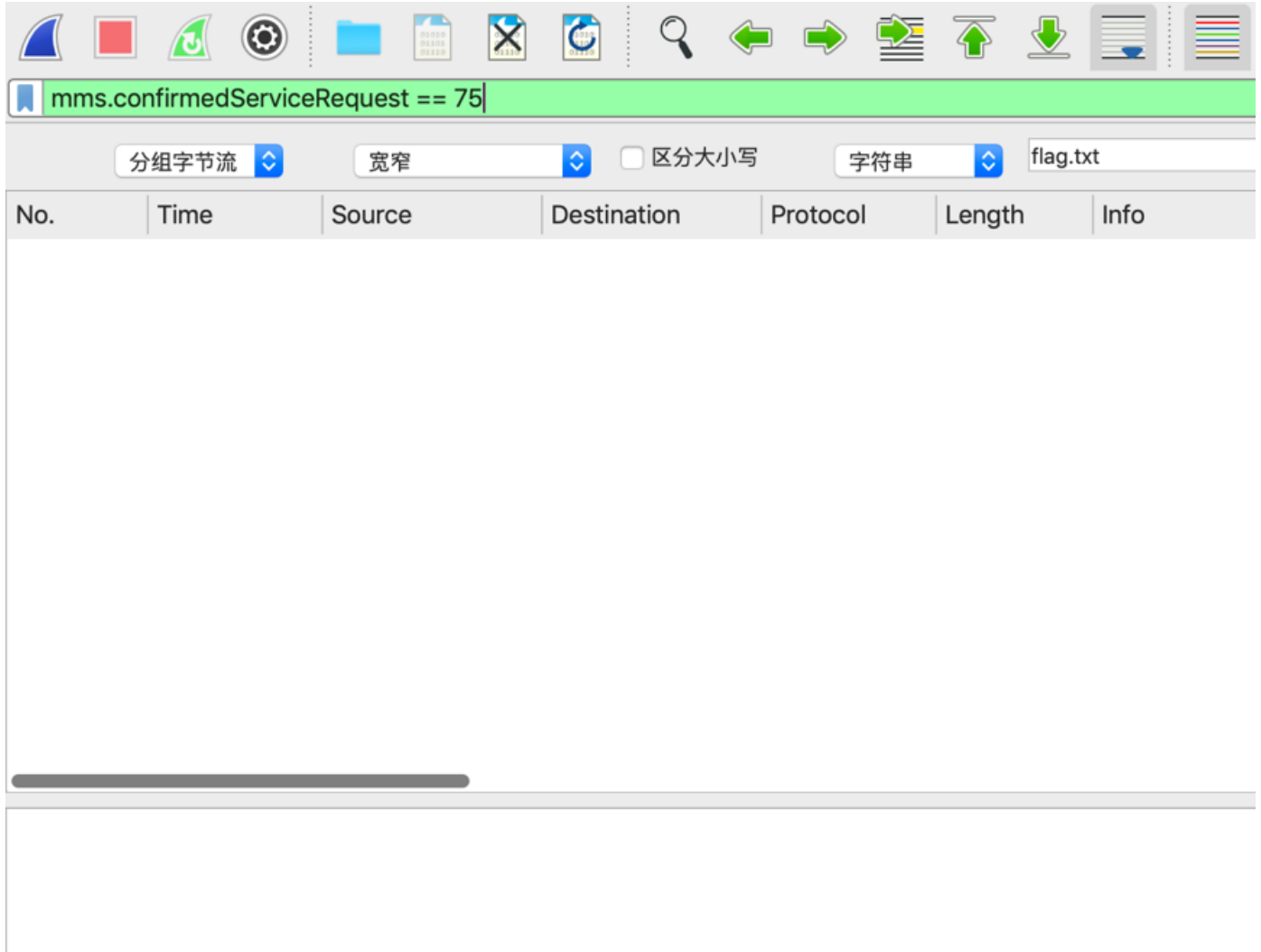
- ▶ Frame 1660: 99 bytes on wire (792 bits), 99 bytes captured (792 bits)
- ▶ Ethernet II, Src: Vmware_b1:29:93 (00:0c:29:b1:29:93), Dst: Vmware_1e:c2:d5 (00:0c:29
- ▶ Internet Protocol Version 4, Src: 192.168.2.112, Dst: 192.168.2.53
- ▶ Transmission Control Protocol, Src Port: 2817, Dst Port: 102, Seq: 4971, Ack: 1037443
- ▶ TPKT, Version: 3, Length: 33
- ▶ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
- ▶ ISO 8327-1 OSI Session Protocol
- ▶ ISO 8327-1 OSI Session Protocol
- ▶ ISO 8823 OSI Presentation Protocol
- ▼ MMS

- ▼ confirmed-RequestPDU
 - invokeID: 507
 - ▼ confirmedServiceRequest: fileRead (73)
 - fileRead: 87502684

No.	Time	Source	Destination	Protocol	Length	Info
1760	55.076134	192.168.2.112	192.168.2.53	MMS	99	508 confirmed-RequestPDU
1802	150.472493	192.168.2.112	192.168.2.53	MMS	99	528 confirmed-RequestPDU
4417	380.291930	192.168.2.112	192.168.2.53	MMS	99	1151 confirmed-RequestPDU
4512	382.365096	192.168.2.112	192.168.2.53	MMS	99	1154 confirmed-RequestPDU
4932	383.144938	192.168.2.112	192.168.2.53	MMS	99	1159 confirmed-RequestPDU
4951	386.034230	192.168.2.112	192.168.2.53	MMS	99	1162 confirmed-RequestPDU
6436	389.256138	192.168.2.112	192.168.2.53	MMS	99	1174 confirmed-RequestPDU
7713	391.790178	192.168.2.112	192.168.2.53	MMS	99	1185 confirmed-RequestPDU
8511	393.843760	192.168.2.112	192.168.2.53	MMS	99	1193 confirmed-RequestPDU

```

▶ Frame 1760: 99 bytes on wire (792 bits), 99 bytes captured (792 bits)
▶ Ethernet II, Src: Vmware_b1:29:93 (00:0c:29:b1:29:93), Dst: Vmware_1e:c2:d5 (00:0c:29:1e:c2:d5)
▶ Internet Protocol Version 4, Src: 192.168.2.112, Dst: 192.168.2.53
▶ Transmission Control Protocol, Src Port: 2817, Dst Port: 102, Seq: 5004, Ack: 1101435, Len: 33
▶ TPKT, Version: 3, Length: 33
▶ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
▶ ISO 8327-1 OSI Session Protocol
▶ ISO 8327-1 OSI Session Protocol
▶ ISO 8823 OSI Presentation Protocol
▼ MMS
  ▼ confirmed-RequestPDU
    invokeID: 508
    confirmedServiceRequest: fileClose (74)
    fileClose: 07502604
  
```



73 为 fileRead 没错，但是 74 为 fileClose ， 75 返回的内容为空，即说明不存在 write 的操作。

那么简单明了，做过滤器 `mms.confirmedServiceRequest == 73`，且在 1771 行后寻找读 `flag.txt` 的 `filedata`

mms.confirmedServiceRequest == 73

分组字节流 宽窄 区分大小写 字符串 flag.txt

No.	Time	Source	Destination	Protocol	Length	Info
300	54.543475	192.168.2.112	192.168.2.53	MMS	99	498 confirmed-RequestPDU
451	54.618571	192.168.2.112	192.168.2.53	MMS	99	499 confirmed-RequestPDU
602	54.662412	192.168.2.112	192.168.2.53	MMS	99	500 confirmed-RequestPDU
754	54.707352	192.168.2.112	192.168.2.53	MMS	99	501 confirmed-RequestPDU
905	54.768299	192.168.2.112	192.168.2.53	MMS	99	502 confirmed-RequestPDU
1055	54.812376	192.168.2.112	192.168.2.53	MMS	99	503 confirmed-RequestPDU
1208	54.866546	192.168.2.112	192.168.2.53	MMS	99	504 confirmed-RequestPDU
1358	54.906909	192.168.2.112	192.168.2.53	MMS	99	505 confirmed-RequestPDU
1510	54.958874	192.168.2.112	192.168.2.53	MMS	99	506 confirmed-RequestPDU
1660	55.019700	192.168.2.112	192.168.2.53	MMS	99	507 confirmed-RequestPDU
1800	150.419434	192.168.2.112	192.168.2.53	MMS	99	527 confirmed-RequestPDU
4005	380.143719	192.168.2.112	192.168.2.53	MMS	99	1148 confirmed-RequestPDU
4216	380.222227	192.168.2.112	192.168.2.53	MMS	99	1149 confirmed-RequestPDU
4369	380.259169	192.168.2.112	192.168.2.53	MMS	99	1150 confirmed-RequestPDU
4422	382.295733	192.168.2.112	192.168.2.53	MMS	99	1153 confirmed-RequestPDU
4517	382.949830	192.168.2.112	192.168.2.53	MMS	99	1156 confirmed-RequestPDU

▶ Frame 1800: 99 bytes on wire (792 bits), 99 bytes captured (792 bits)

▶ Ethernet II, Src: Vmware_b1:29:93 (00:0c:29:b1:29:93), Dst: Vmware_1e:c2:d5 (00:0c:29:1e:c2:d5)

▶ Internet Protocol Version 4, Src: 192.168.2.112, Dst: 192.168.2.53

▶ Transmission Control Protocol, Src Port: 2817, Dst Port: 102, Seq: 5873, Ack: 1103156, Len: 33

▶ TPKT, Version: 3, Length: 33

▶ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol

▶ ISO 8327-1 OSI Session Protocol

▶ ISO 8327-1 OSI Session Protocol

▶ ISO 8823 OSI Presentation Protocol

▼ MMS

- confirmed-RequestPDU
 - invokeID: 527
 - confirmedServiceRequest: fileRead (73)
 - fileRead: 87504092

```

0000  00 0c 29 1e c2 d5 00 0c 29 b1 29 93 08 00 45 00  ..).....).....E.
0010  00 55 87 f8 40 00 40 06 2c b5 c0 a8 02 70 c0 a8  .U..@.@.,...p..
0020  02 35 0b 01 00 66 c9 75 64 1a 44 69 3a a8 80 18  .5...f.u d.Di:..
0030  fa b5 4d 0b 00 00 01 01 08 0a 00 0b 64 54 00 09  .M.....dT...
0040  e4 14 03 00 00 21 02 f0 80 01 00 01 00 61 14 30  ....!.....a.0
0050  12 02 01 03 a0 0d a0 0b 02 02 02 0f 9f 49 04 05  .....I..
0060  37 34 dc                                     74.

```

发现在 1771 行后最近的一行为 1800，这行为 RequestPDU，于是找到其对应 invokeID=527 的 ResponsePDU:

The screenshot shows a Wireshark capture of MMS traffic. The packet list pane displays two packets:

No.	Time	Source	Destination	Protocol	Length	Info
1800	150.419434	192.168.2.112	192.168.2.53	MMS	99	527 confirmed-RequestPDU
1801	150.466006	192.168.2.53	192.168.2.112	MMS	109	527 confirmed-ResponsePDU

The packet details pane for the selected packet (1801) shows the following structure:

- Frame 1801: 109 bytes on wire (872 bits), 109 bytes captured (872 bits)
- Ethernet II, Src: Apple_c7:88:1e (60:f8:1d:c7:88:1e), Dst: Vmware_b1:29:93 (00:0c:29:b1:29:93)
- Internet Protocol Version 4, Src: 192.168.2.53, Dst: 192.168.2.112
- Transmission Control Protocol, Src Port: 102, Dst Port: 2817, Seq: 1103156, Ack: 5906, Len: 43
- TPKT, Version: 3, Length: 43
- ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
- ISO 8327-1 OSI Session Protocol
- ISO 8327-1 OSI Session Protocol
- ISO 8823 OSI Presentation Protocol
- MMS
 - confirmed-ResponsePDU
 - invokeID: 527
 - confirmedServiceResponse: fileRead (73)
 - fileRead
 - fileData: 363138353040313032
 - moreFollows: False

The packet bytes pane shows the raw data in hexadecimal and ASCII. The ASCII representation of the file data is highlighted in red:

```

0000 00 0c 29 b1 29 93 60 f8 1d c7 88 1e 08 00 45 00  ..).).....E
0010 00 5f 1b f6 40 00 80 06 58 ad c0 a8 02 35 c0 a8  _@...X...S
0020 02 70 08 66 0b 01 44 69 3a a8 c9 75 64 3b 80 18  p.f-Di:ud;
0030 01 04 b9 25 00 00 01 01 08 0a 00 09 e4 19 00 0b  .%.....
0040 64 54 03 00 00 2b 02 f0 80 01 00 01 00 61 1e 30  dT...+...a 0
0050 1c 02 01 03 a0 17 a1 15 02 02 02 0f bf 49 0e 80  I...
0060 09 36 31 38 35 30 40 31 30 32 81 01 00          61850@1 02
  
```

根据结果，发现其 fileData 内容为：363138353040313032

对应的 ASCII 内容为：61850@102

即是最终的 flag

0x04 总结

此题大佬的解法很简单，只要知道对应的MMS协议的规约中Confirmed-RequestPDU和Confirmed-ResponsePDU的结构然后找到对应的服务，直接一个脚本搞定。

如下：

```

import pyshark

def flag():
    try:
        captures = pyshark.FileCapture("question_1531222544_JYvFGmLP49PFC0R2.pcap")
        flag_frsm = False
        flag_frsm_id = None
        flag_read = False
        for capture in captures:
            for pkt in capture:
                if pkt.layer_name == "mms":
                    # file open
                    if hasattr(pkt, "confirmedservicerequest") and int(pkt.confirmedservicerequest) == 72:
                        if hasattr(pkt, "filename_item"):
                            filename_items = pkt.filename_item.fields
                            for f in filename_items:
                                file_name = str(f.get_default_value())
                                if file_name == "flag.txt":
                                    flag_frsm = True
                    if hasattr(pkt, "confirmedservicerresponse") and int(pkt.confirmedservicerresponse) == 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") and int(pkt.confirmedservicerequest) == 73 and fl
ag_frsm_id:
                            if hasattr(pkt, "fileread"):
                                if str(pkt.fileread) == str(flag_frsm_id):
                                    flag_read = True
                                    flag_frsm_id = None
                                if hasattr(pkt, "confirmedservicerresponse") and int(pkt.confirmedservicerresponse) == 73 and
flag_read:
                                    if hasattr(pkt, "filedata"):
                                        data = str(pkt.filedata).replace(":", "")
                                        print(hex_to_ascii(data))
                                    flag_read = False
                    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()

```

工控协议有很多，代表性的除了 MMS 外还有 Modbus、DNP3、Melsec-Q、S7、Ethernet/IP 等，

出题的方式也有很多，主要还是对这个协议比较熟悉，那么解什么题都很容易了

刚研究工控安全，如果文字描述有误，还望斧正，另外，如果有兴趣的小伙伴可以一起研究工控安全的内容~

0x04 参考

MMS报文分析示例

2018年工业信息安全技能大赛（东北赛区）解题报告——工业网络数据分析

ICS_CTF Contest

=====

传送门:

链接:https://pan.baidu.com/s/1wL_1fErCAwQIF_DYPIOiQ

密码:9mb8



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)