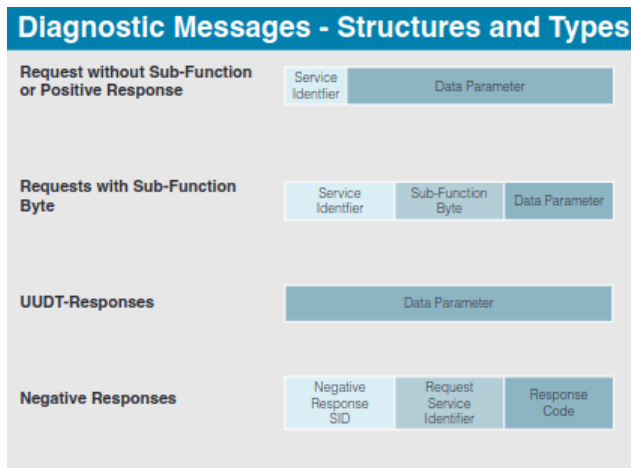
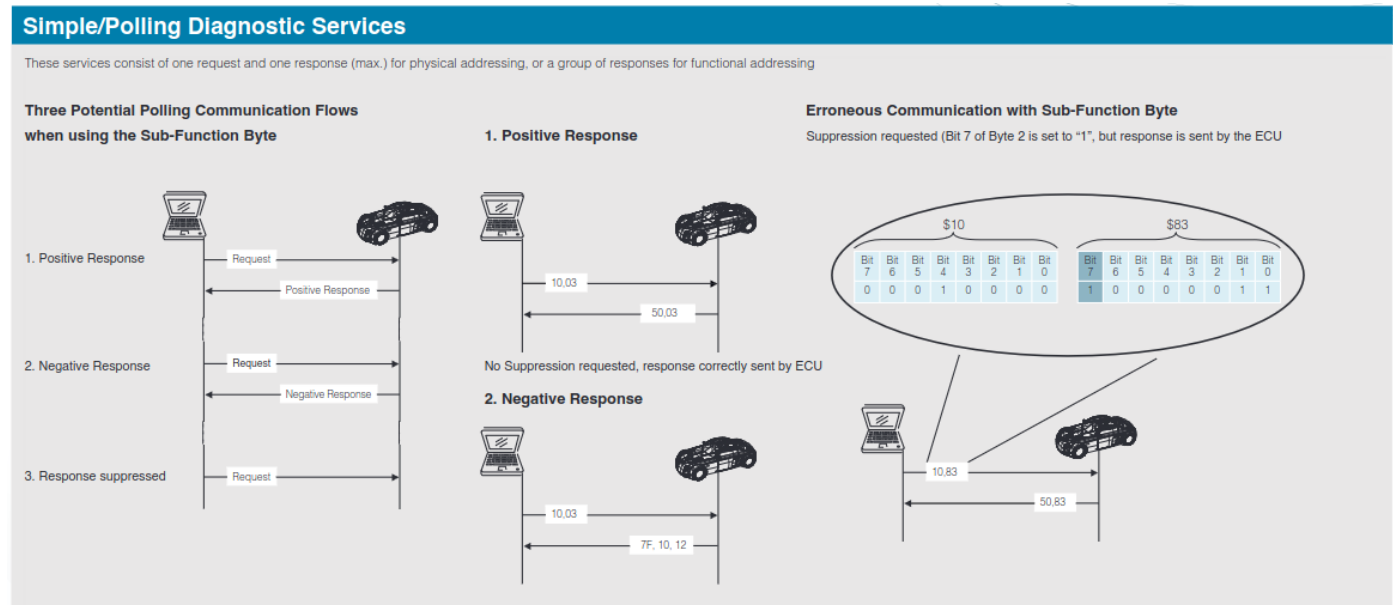


16.4.

https://automotive.softing.com/fileadmin/sof-files/pdf/de/ae/poster/UDS_Faltposter_softing2016.pdf

- [UDS_Faltposter_softing2016.pdf](#)



Izgleda da Scapy vec ima implementiran i UDS protokol

17.4.

<https://munich.dissec.to/kb/chapters/uds/uds-scapy.html>

<https://scapy.readthedocs.io/en/latest/layers/automotive.html#layers>

koristenje:

```
> echo "63 42" | isotpsocket -s 101 -d 100 vcan0
> echo "23 42" | isotpsocket -s 101 -d 100 vcan0
```

```
>>> s = ISOTPNativeSocket(iface="vcan0", tx_id=0x100, rx_id=0x101, basecls=UDS)
>>> while True:
...     s.recv()
```

```
...
<UDS service=ReadMemoryByAddressPositiveResponse |<UDS_RMBAPR dataRecord='B' |>>
<UDS service=ReadMemoryByAddress |<UDS_RMBA memorySizeLen=4 memoryAddressLen=2 |>>
```

zadatak za Hackultet

1. opcija

- napraviti mali binary sa zastavicom koji bi trebao biti primjer ECU programa
- napraviti ga dosutpnim putem UDS ReadMemoryByAddress servisa
- natjecatelj mora odskenirati ECU s caringcaribouom ili scapyem, i iskoristiti ReadMemoryByAddress servis da dohvati binary
 - parametre za read memory by address (32 bitna adresa) moze iscitati iz zadatka ili nekog identifera
- binary potom treba reverzati i iz njega izvuci flag

2. opcija

- potrebno je koristeci ReadDataByIdentifier saznati koji nesigurni algoritam se koristi za Authentication servis (primjerice SHA2 se potpuno krivo koristi za generiranje kljuka iz dobivenog seed-a)
 - ovo se mozda moze iscitati i iz drugog ECU-a
- prva zastavica moze se dohvatiti iz nekog drugog identifera
- nakon autentificiranja korisnik moze koristiti ReadMemoryByAddress kao u prvom zadatku da dohvati program ECU-a iz memorije i iz njega izvuce drugu zastavicu

scapy ReadMemoryByAddress

```
class UDS_RMBA(Packet):
    name = 'ReadMemoryByAddress'
    fields_desc = [
        BitField(name='memorySizeLen', default=0, size=4),
        BitField(name='memoryAddressLen', default=0, size=4),
        ConditionalField(XByteField(name='memoryAddress1', default=0),
            lambda pkt: pkt.memoryAddressLen == 1),
        ConditionalField(XShortField(name='memoryAddress2', default=0),
            lambda pkt: pkt.memoryAddressLen == 2),
        ConditionalField(X3BytesField(name='memoryAddress3', default=0),
            lambda pkt: pkt.memoryAddressLen == 3),
        ConditionalField(XIntField(name='memoryAddress4', default=0),
            lambda pkt: pkt.memoryAddressLen == 4),
        ConditionalField(XByteField(name='memorySize1', default=0),
            lambda pkt: pkt.memorySizeLen == 1),
        ConditionalField(XShortField(name='memorySize2', default=0),
            lambda pkt: pkt.memorySizeLen == 2),
        ConditionalField(X3BytesField(name='memorySize3', default=0),
            lambda pkt: pkt.memorySizeLen == 3),
        ConditionalField(XIntField(name='memorySize4', default=0),
            lambda pkt: pkt.memorySizeLen == 4),
```

18.4.

scapy prepoznaje je li neki paket odgovor na neki drugi koristeći answers metodu svakog paketa, primjerice je li dobiveni UDS_ERPR paket odgovor na poslani UDS_PR:

```

class UDS_ERPR(Packet):
    name = 'ECUResetPositiveResponse'
    fields_desc = [
        ByteEnumField(name='resetType', default=0, UDS_ER.resetTypes),
        ConditionalField(ByteField(name='powerDownTime', default=0),
                        lambda pkt: pkt.resetType == 0x04)
    ]

    def answers(self, other):
        return isinstance(other, UDS_ER) and other.resetType == self.resetType

```

da bi ovo ispravno radilo, potrebno je socketu specificirati basecls paketa, odnosno aplikacijski sloj, u našem slučaju isotp socketu je potrebno specificirati UDS kao basecls, jer će koristiti UDS.answers kao početnu metodu iz koje se kasnije poziva answers određenog tipa paketa, primjerice UDS_ERPR.answers:

```

class UDS(ISOTP):
    name = 'UDS'
    fields_desc = [
        XByteEnumField(name='service', default=0, services)
    ]

    def answers(self, other):
        # type: (Union[UDS, Packet]) -> bool
        if other.__class__ != self.__class__:
            return False
        if self.service == 0x7f:
            return self.payload.answers(other)
        if self.service == (other.service + 0x40):
            if isinstance(self.payload, NoPayload) or \
                isinstance(other.payload, NoPayload):
                return len(self) <= len(other)
            else:
                return self.payload.answers(other.payload)
        return False

```

19.4.

Prekoderao sve iz python-can-a u scapy.