

HackGDL 2026 - Pueblito Mágico del Car Hacking CTF

1) CAN

1.1) Transmission

1. Very easy: There is a flag broadcast with CAN ID 0x770 if you send any message with standard ID 0x456.
2. Very easy: There is a flag broadcast with CAN ID 0x771 if you send a CAN message with standard ID 0x457 and payload "GIVEFLAG".
3. Very easy: There is a flag broadcast with CAN ID 0x772 if you send a remote frame with a specific standard ID.
4. Intermediate: There is a flag broadcast with CAN ID 0x773 if you send a specific CAN message, checked with the following function:

```
static uint8_t checkIfShouldSendFlag4(const FDCAN_RxHeaderTypeDef* pHeader, const
uint8_t* data)
{
    8001ccc: b480      push {r7}
    8001cce: b083      sub sp, #12
    8001cd0: af00      add r7, sp, #0
    8001cd2: 6078      str r0, [r7, #4]
    8001cd4: 6039      str r1, [r7, #0]
    8001cd6: 687b      ldr r3, [r7, #4]
    8001cd8: 689b      ldr r3, [r3, #8]
    8001cda: 2b00      cmp r3, #0
    8001cdc: d12d      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
    8001cde: 687b      ldr r3, [r7, #4]
    8001ce0: 681b      ldr r3, [r3, #0]
    8001ce2: 4a19      ldr r2, [pc, #100] @ (8001d48 <checkIfShouldSendFlag4+0x7c>)
    8001ce4: 4293      cmp r3, r2
    8001ce6: d128      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
    8001ce8: 683b      ldr r3, [r7, #0]
    8001cea: 781b      ldrb r3, [r3, #0]
    8001cec: 2b50      cmp r3, #80 @ 0x50
    8001cee: d124      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
    8001cf0: 683b      ldr r3, [r7, #0]
    8001cf2: 3301      adds r3, #1
    8001cf4: 781b      ldrb r3, [r3, #0]
    8001cf6: 2b34      cmp r3, #52 @ 0x34
    8001cf8: d11f      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
    8001cfa: 683b      ldr r3, [r7, #0]
    8001cfc: 3302      adds r3, #2
}
```

```

8001cfe: 781b      ldrb r3, [r3, #0]
8001d00: 2b24      cmp r3, #36 @ 0x24
8001d02: d11a      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
8001d04: 683b      ldr r3, [r7, #0]
8001d06: 3303      adds r3, #3
8001d08: 781b      ldrb r3, [r3, #0]
8001d0a: 2b24      cmp r3, #36 @ 0x24
8001d0c: d115      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
8001d0e: 683b      ldr r3, [r7, #0]
8001d10: 3304      adds r3, #4
8001d12: 781b      ldrb r3, [r3, #0]
8001d14: 2b57      cmp r3, #87 @ 0x57
8001d16: d110      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
8001d18: 683b      ldr r3, [r7, #0]
8001d1a: 3305      adds r3, #5
8001d1c: 781b      ldrb r3, [r3, #0]
8001d1e: 2b30      cmp r3, #48 @ 0x30
8001d20: d10b      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
8001d22: 683b      ldr r3, [r7, #0]
8001d24: 3306      adds r3, #6
8001d26: 781b      ldrb r3, [r3, #0]
8001d28: 2b52      cmp r3, #82 @ 0x52
8001d2a: d106      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
8001d2c: 683b      ldr r3, [r7, #0]
8001d2e: 3307      adds r3, #7
8001d30: 781b      ldrb r3, [r3, #0]
8001d32: 2b44      cmp r3, #68 @ 0x44
8001d34: d101      bne.n 8001d3a <checkIfShouldSendFlag4+0x6e>
8001d36: 2301      movs r3, #1
8001d38: e000      b.n 8001d3c <checkIfShouldSendFlag4+0x70>
8001d3a: 2300      movs r3, #0
8001d3c: 4618      mov r0, r3
8001d3e: 370c      adds r7, #12
8001d40: 46bd      mov sp, r7
8001d42: f85d 7b04 ldr.w r7, [sp], #4
8001d46: 4770      bx lr
8001d48: 0077cafe .word 0x0077cafe

```

1.2) Reception

Send a message with standard ID 0x458 and an empty payload to enable the periodic transmission of the following flags.

1. Easy: There is a flag broadcast one bit at a time with CAN ID 0x6F0.
2. Intermediate: There is a flag broadcast one bit at a time with CAN ID 0x6F1.
3. Difficult: There is a flag broadcast one bit at a time with CAN ID 0x6F2 (flag is 19 characters long).

2) UDS

1. Very easy: ECU D holds a flag readable at ID 0x0001 with the Read Data By Identifier Service.
2. Easy: ECU D holds a flag readable at ID 0x0002, but you will need to authenticate first.
3. Easy (requires scripting): ECU D holds another flag readable with the Read Data By Identifier Service.
4. Easy: ECU D holds a flag that is accessible with Read Memory By Address at address 0x01234567 and size 17 (0x11).
5. Intermediate (requires scripting): ECU D has a custom UDS service with ID 0x40, can you create a valid request?
6. Exploitation (Advanced)

There are two UDS routines that implement a vulnerable password check, and which require slightly different exploitation approaches. Can you exploit them to recover all the previous flags?

The goal is not to recover the password or bypass the check. The goal is to overtake execution in order to dump memory (over USB, CAN, etc.). Check Writing ARM shell code for RAMN for guidance on how to get started. You can compile the source code yourself to generate .elf and .map files. Although a JTAG debugger isn't absolutely necessary, it will make exploitation easier. You will need to use the Routine Control service after performing a Security Access (see Interacting with UDS). If you are already familiar with ARM exploitation, you may add yourself extra constraints that are not enforced (such as not executing any code in RAM).

Routine Control 0x20A

```
const char expected_password[] = "VULNERABILITY";
static void RAMN_UDS_RoutineControlVulnerabilityExample(uint8_t* data, uint16_t size)
{
    if (checkAuthenticated() == True)
    {
        char stack_password[sizeof(expected_password)];

        if ((size > 4U) && (size < ISOTP_RXBUFFER_SIZE))
        {
            data[size] = 0U; //zero-terminate buffer
            strcpy(stack_password, (char*)&data[4U]); //copy string
            if (strcmp(stack_password, expected_password) == 0U)
                RAMN_UDS_FormatPositiveResponseEcho(data, 4U);
            else RAMN_UDS_FormatNegativeResponse(data, UDS_NRC_ROOR);
        }
        else RAMN_UDS_FormatNegativeResponse(data, UDS_NRC_IMLOIF);
    }
    else RAMN_UDS_FormatNegativeResponse(data, UDS_NRC_SAD);
}
```

Routine Control 0x20B

```
const char expected_password[] = "VULNERABILITY";
char ctf_global_password[sizeof(expected_password)];
static void RAMN_UDS_RoutineControlVulnerabilityExample2(uint8_t* data, uint16_t size)
{
    if (checkAuthenticated() == True)
    {
        if ((size > 4U) && (size < ISOTP_RXBUFFER_SIZE))
        {
            data[size] = 0U; //zero-terminate buffer
            strcpy(ctf_global_password, (char*)&data[4U]); //copy string
            if (strcmp(ctf_global_password, expected_password) == 0U)
                RAMN_UDS_FormatPositiveResponseEcho(data, 4U);
            else RAMN_UDS_FormatNegativeResponse(data, UDS_NRC_ROOR);
        }
        else RAMN_UDS_FormatNegativeResponse(data, UDS_NRC_IMLOIF);
    }
    else RAMN_UDS_FormatNegativeResponse(data, UDS_NRC_SAD);
}
```