

# DJI Reverse Engineering

# About Us

ASTRONAUT (astronaut.am): The company's initial focus was on electronics education, but it subsequently established a separate (unofficial) research team.

- ArmSec 2024: Reverse Engineering: UEFI BIOS Malwares
- CyberGen 2025: First time

Hrant Tadevosyan

- LinkedIn: <https://am.linkedin.com/in/hrant-tadevosyan-a75741200>
- Email: [hrant.tadevosyan@protonmail.com](mailto:hrant.tadevosyan@protonmail.com)

Levon Martirosyan\*

- LinkedIn: <https://www.linkedin.com/in/levon-martirosyan-b98251189>
- Email: [levonmartirosyan2019@gmail.com](mailto:levonmartirosyan2019@gmail.com)

# Motivation

DJI is the preeminent drone company, with applications in both military and civilian sectors.

## Security:

- Proprietary code can't be trusted
- Information presented in research papers must be validated
- Hunting for vulnerabilities

## Modifications:

- Extending capabilities
- Removing limitations

# Agenda

## Software Analysis:

- DJI Assistant 2 introduction
- Network monitoring
- USB interactions

## Firmware Analysis:

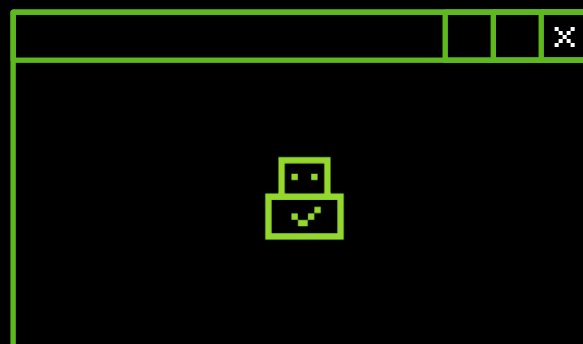
- File format details
- Packing algorithm

## Hardware Analysis:

- Internal circuitry
- Component descriptions

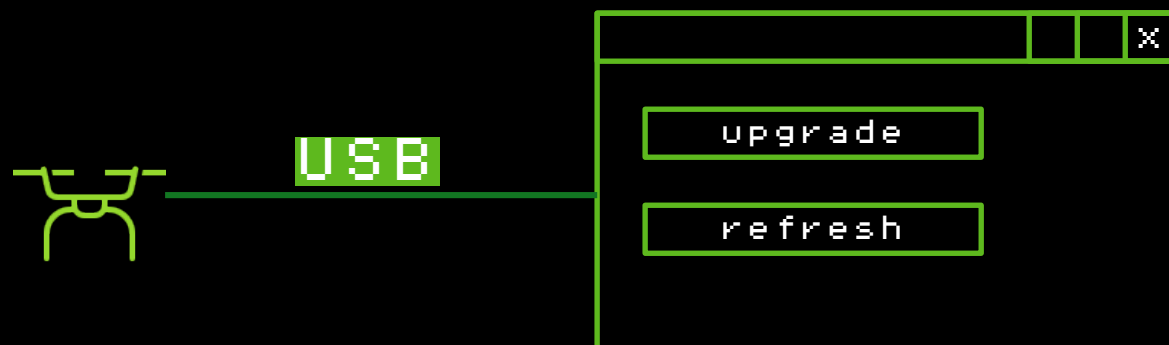
# DJI Assistant 2

- It is used to interact with the hardware
- It is the only official way
- No direct links for downloading the firmware
- Works only on Windows and MacOS\*



# DJI Assistant 2

- It is used to interact with the hardware
- It is the only official way
- No direct links for downloading the firmware
- Works only on Windows and MacOS\*



# DJI Assistant 2 (Files)

```
> dir *.* /s
```

```
1132
```

```
> dir
```

```
<DIR>
```

```
<DIR>
```

```
182,272 DJI Assistant 2.exe
```

```
<DIR>
```

```
DJIApp
```

```
<DIR>
```

```
DJIEngine
```

```
<DIR>
```

```
Drivers
```

```
449,816 msvcp140.dll
```

```
5,298,296 Qt5Core.dll
```

```
296,647 unins000.dat
```

```
3,252,285 unins000.exe
```

```
83,224 voruntime140.dll
```

# DJI Assitant 2.exe

```
...
1 push     offset aDjiservicecore ; "DJIServiceCore.exe"
2 call     esi ; QString::fromAscii_helper(char const *,int)
   mov     [ebp+var_14], eax
   lea     eax, [ebp+var_14]
   mov     byte ptr [ebp+var_4], 2
   push    eax
3 call     re_kill_proc
...
```

\*NOTE that `re_` prefixed labels have been manually reverse engineered



# re\_kill\_proc

```
...
push    offset aDjiservicecore ; "DJIServiceCore.exe"
call    esi ; QString::fromAscii_helper(char const *,int)
mov     lebp+var_141, eax
lea     eax, [ebp+var_141]
mov     byte ptr [ebp+var_41], 2
push    eax
call    re_kill_proc
...
```

```
re_kill_proc:
...
push    0
lea     ecx, [ebp+var_241]
1  call    ds:___imp_QProcess::QProcess
push    8
2  push    offset aPkill1 ; "pkill %1"
mov     [ebp+var_41], 0
3  call    ds:___imp_QString::fromAscii_helper
...
push    eax
mov     dword ptr [ecx], 3
lea     ecx, [ebp+var_241]
4  call    ds:___imp_QProcess::start
...
```

# main

```
...
push    offset aDjIServiceCore ; "DJIServiceCore.exe"
call    esi ; QString::fromAscii_helper(char const *,int)
mov     [ebp+var_14], eax
lea     eax, [ebp+var_14]
mov     byte ptr [ebp+var_4], 2
push    eax
call    re_kill_proc
...
1  lea     eax, [ebp+var_10]
push    offset aDjIServiceExe ; "DJIService.exe"
2  push    eax
call    ds:__imp_QString::fromUtf8
add     esp, 0Ch
push    eax
lea     ecx, [ebp+var_14]
...
3  mov     byte ptr [ebp+var_4], 0Ah
call    ds:__imp_QString::append
...
push    0
lea     eax, [ebp+var_10]
push    eax
push    edi
push    esi
4  call    ds:__imp_QProcess::startDetached
...
```

## DJIService.exe (IAT)

Address	Name
0x00A727D0	usb_init
0x00A727CC	usb_set_debug
0x00A727C0	usb_find_busses
0x00A727D4	usb_find_devices
0x00A727D8	usb_get_busses
...	...

The library is inside the install folder, called `libusb0_dji.dll`

## DJIService.exe (libusb0)

```
...  
call    usb_init  
push    4  
call    usb_set_debug  
add     esp, 4  
call    usb_find_busses  
call    usb_find_devices  
call    usb_get_busses  
...
```

We'll show only the internals of **one** function. Other functions listed here operate similarly.

# libusb0\_dji.dll

```
usb_init:
    .text
1  mov     edi, offset "USB_DEBUG"
   push   edi
2  call    esi ds:getenv
   push   eax
   call    ds:atoi
   push   eax
3  call    usb_set_debug
    .text
4  jmp     re_os_usb_init
```

- Uses **USB\_DEBUG** environment variable to adjust the debug level
- Calls an internal function to do the rest

# re\_os\_usb\_init

## re\_os\_usb\_init:

```
...  
1 push    [ebp+re_loop_index]  
2 lea     eax, [ebp+re_path]  
  push    offset "\\.\libusb0-"  
  push    offset "%s%04d"  
  push    1FFh  
  push    eax  
3 call    ds:__snprintf  
...  
4 lea     eax, [ebp+re_path]  
  push    eax  
5 call    ds:CreateFileA  
6 mov     [ebp+re_dev_hnd], eax  
  cmp     eax, 0FFFFFFFFh  
7 jz      short loc_re_rewind
```

## loc\_re\_os\_usb\_init\_ioctl:

```
8  push    18h  
  lea     ecx, [ebp+re_version]  
  push    ecx  
  push    18h  
  push    ecx  
9  push    222048h  
  push    eax  
  lea     ebx, [ebp+re_bytes_tx]  
10 call    re_ioctl  
...
```

# re\_os\_usb\_init

## re\_os\_usb\_init:

```
...  
1 push    [ebp+re_loop_index]  
2 lea     eax, [ebp+re_path]  
  push    offset "\\.\libusb0-"  
  push    offset "%s%04d"  
  push    1FFh  
  push    eax  
3 call    ds:__snprintf  
...  
4 lea     eax, [ebp+re_path]  
  push    eax  
5 call    ds:CreateFileA  
6 mov     [ebp+re_dev_hnd], eax  
  cmp     eax, 0FFFFFFFFh  
7 jz      short loc_re_rewind
```

## loc\_re\_os\_usb\_init\_ioctl:

```
8  push    18h  
  lea     ecx, [ebp+re_version]  
  push    ecx  
  push    18h  
  push    ecx  
9  push    222048h  
  push    eax  
  lea     ebx, [ebp+re_bytes_tx]  
10 call    re_ioctl  
...
```

3. Prepares a `\\.\libusb0-X` string
5. Tries to open the file
8. If it exists, then creates a `0x18` long buffer
9. Stores an unknown value `0x222048`
10. Calls an internal function, called `re_ioctl`

# re\_ioctl

```
re_ioctl:
    ...
    lea     eax, [ebp+Overlapped]
    push    eax
    push    esi
    push    [ebp+nOutBufferSize]
    push    [ebp+lpOutBuffer]
    push    [ebp+nInBufferSize]
    push    [ebp+lpInBuffer]
    push    [ebp+dwIoControlCode]
    push    [ebp+hDevice]
    call    ds:DeviceIoControl
    ...
```

- The device path is `\\.\libusb0-X` string
- Data structure is `0x18` bytes long
- The IOCTL code is `0x222048`



# re\_ioctl

```
re_ioctl:
    ...
    lea     eax, [ebp+Overlapped]
    push    eax
    push    esi
    push    [ebp+nOutBufferSize]
    push    [ebp+lpOutBuffer]
    push    [ebp+nInBufferSize]
    push    [ebp+lpInBuffer]
    push    [ebp+dwIoControlCode]
    push    [ebp+hDevice]
    call    ds:DeviceIoControl
    ...
```

- The device path is `\\.\libusb0-X` string
- Data structure is `0x18` bytes long
- The IOCTL code is `0x222048`
  - Device: (code >> 16) 8 0xFFFF
  - Access: (code >> 14) 8 3
  - Function: (code >> 2) 8 0xFFF
  - Method: code 8 3

# re\_ioctl

```
re_ioctl:
    ...
    lea     eax, [ebp+Overlapped]
    push    eax
    push    esi
    push    [ebp+nOutBufferSize]
    push    [ebp+lpOutBuffer]
    push    [ebp+nInBufferSize]
    push    [ebp+lpInBuffer]
    push    [ebp+dwIoControlCode]
    push    [ebp+hDevice]
    call    ds:DeviceIoControl
    ...
```

- The device path is `\\.\libusb0-X` string
- Data structure is `0x18` bytes long
- The IOCTL code is `0x222048`
  - Device: `(0x222048 >> 16) & 0xFFFF`
  - Access: `(0x222048 >> 14) & 3`
  - Function: `(0x222048 >> 2) & 0xFFF`
  - Method: `0x222048 & 3`

# re\_ioctl

```
re_ioctl:
    ...
    lea     eax, [ebp+Overlapped]
    push    eax
    push    esi
    push    [ebp+nOutBufferSize]
    push    [ebp+lpOutBuffer]
    push    [ebp+nInBufferSize]
    push    [ebp+lpInBuffer]
    push    [ebp+dwIoControlCode]
    push    [ebp+hDevice]
    call    ds:DeviceIoControl
    ...
```

- The device path is `\\.\libusb0-X` string
- Data structure is `0x18` bytes long
- The IOCTL code is `0x222048`
  - Device: `0x022`
  - Access: `0x000`
  - Function: `0x812`
  - Method: `0x000`

# re\_ioctl

```
re_ioctl:
    ...
    lea     eax, [ebp+Overlapped]
    push    eax
    push    esi
    push    [ebp+nOutBufferSize]
    push    [ebp+lpOutBuffer]
    push    [ebp+nInBufferSize]
    push    [ebp+lpInBuffer]
    push    [ebp+dwIoControlCode]
    push    [ebp+hDevice]
    call    ds:DeviceIoControl
    ...
```

- The device path is `\\.\libusb0-X` string
- Data structure is `0x18` bytes long
- The IOCTL code is `0x222048`
  - Device: `0x022` = `FILE_DEVICE_UNKNOWN`
  - Access: `0x000` = `FILE_ANY_ACCESS`
  - Function: `0x812` = `re_get_version`
  - Method: `0x000` = `METHOD_BUFFERED`

# Recap

DJI Assistant 2.exe

# Recap

DJI Assistant 2.exe

DJIService.exe

# Recap

DJI Assistant 2.exe

DJIService.exe

libusb0\_dji.dll

# Recap

DJI Assistant 2.exe

DJIService.exe

libusb0\_dji.dll

Device: libusb0-X

?Driver.sys?



# Recap

DJI Assistant 2.exe

DJIService.exe

libusb0\_dji.dll

Device: libusb0-X

libusb0\_device.sys

- Use the command `driverquery`
- Or look inside the installation folder
- Or look inside the System32/drivers folder
- ...

# libusb0\_device.sys

```
NTSTATUS
DriverEntry(
    PDRIVER_OBJECT DriverObject,
    PUNICODE_STRING RegistryPath)
{
    1 re_log("DriverEntry: [loading-driver] v%d.%d.%d.%d\n", 1, 2, 6, 0);
    2 memset64(DriverObject->MajorFunction, &re_major_func, 0x10u);
    3 DriverObject->DriverExtension->AddDevice = re_add_device;
    4 DriverObject->DriverUnload = re_unload;
    return 0;
}
```

\*NOTE that this is a decompiled function, and NOT a snippet from the actual source file.

1. just a log record

# libusb0\_device.sys

```
NTSTATUS
DriverEntry(
    PDRIVER_OBJECT DriverObject,
    PUNICODE_STRING RegistryPath)
{
    1 re_log("DriverEntry: [loading-driver] v%d.%d.%d.%d\n", 1, 2, 6, 0);
    2 memset64(DriverObject->MajorFunction, &re_major_func, 0x10u);
    3 DriverObject->DriverExtension->AddDevice = re_add_device;
    4 DriverObject->DriverUnload = re_unload;
    return 0;
}
```

\*NOTE that this is a decompiled function, and NOT a snippet from the actual source file.

1. just a log record
2. sets the major function pointers:  
DriverObject->MajorFunction[0] = my\_operation0\_function;  
DriverObject->MajorFunction[1] = my\_operation1\_function;  
...

# libusb0\_device.sys

```
NTSTATUS
DriverEntry(
    PDRIVER_OBJECT DriverObject,
    PUNICODE_STRING RegistryPath)
{
    1 re_log("DriverEntry: [loading-driver] v%d.%d.%d.%d\n", 1, 2, 6, 0);
    2 memset64(DriverObject->MajorFunction, &re_major_func, 0x100);
    3 DriverObject->DriverExtension->AddDevice = re_add_device;
    4 DriverObject->DriverUnload = re_unload;
    return 0;
}
```

\*NOTE that this is a decompiled function, and NOT a snippet from the actual source file.

1. just a log record

2. sets the major function pointers:

```
DriverObject->MajorFunction[IRP_MJ_READ] = my_read_function;
```

```
DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] = my_dev_ctl_function;
```

```
...
```

# libusb0\_device.sys

```
NTSTATUS
DriverEntry(
    PDRIVER_OBJECT DriverObject,
    PUNICODE_STRING RegistryPath)
{
    1 re_log("DriverEntry: [loading-driver] v%d.%d.%d.%d\n", 1, 2, 6, 0);
    2 memset64(DriverObject->MajorFunction, &re_major_func, 0x100);
    3 DriverObject->DriverExtension->AddDevice = re_add_device;
    4 DriverObject->DriverUnload = re_unload;
    return 0;
}
```

\*NOTE that this is a decompiled function, and NOT a snippet from the actual source file.

1. just a log record

2. sets the major function pointers:

DriverObject->MajorFunction[IRP\_MJ\_READ] = re\_major\_func;

DriverObject->MajorFunction[IRP\_MJ\_DEVICE\_CONTROL] = re\_major\_func;

...

# libusb0\_device.sys

```
NTSTATUS
DriverEntry(
    PDRIVER_OBJECT DriverObject,
    PUNICODE_STRING RegistryPath)
{
    1 re_log("DriverEntry: [loading-driver] v%d.%d.%d.%d\n", 1, 2, 6, 0);
    2 memset64(DriverObject->MajorFunction, &re_major_func, 0x10u);
    3 DriverObject->DriverExtension->AddDevice = re_add_device;
    4 DriverObject->DriverUnload = re_unload;
    return 0;
}
```

\*NOTE that this is a decompiled function, and NOT a snippet from the actual source file.

1. just a log record
2. sets the major function pointers:  
DriverObject->MajorFunction[IRP\_MJ\_READ] = re\_major\_func;  
DriverObject->MajorFunction[IRP\_MJ\_DEVICE\_CONTROL] = re\_major\_func;  
...
3. adds new devices, most likely \\.\libusb0-X

# libusb0\_device.sys

```
NTSTATUS
DriverEntry(
    PDRIVER_OBJECT DriverObject,
    PUNICODE_STRING RegistryPath)
{
    1 re_log("DriverEntry: [loading-driver] v%d.%d.%d.%d\n", 1, 2, 6, 0);
    2 memset64(DriverObject->MajorFunction, &re_major_func, 0x10u);
    3 DriverObject->DriverExtension->AddDevice = re_add_device;
    4 DriverObject->DriverUnload = re_unload;
    return 0;
}
```

\*NOTE that this is a decompiled function, and NOT a snippet from the actual source file.

1. just a log record
2. sets the major function pointers:  
DriverObject->MajorFunction[IRP\_MJ\_READ] = re\_major\_func;  
DriverObject->MajorFunction[IRP\_MJ\_DEVICE\_CONTROL] = re\_major\_func;  
...
3. adds new devices, most likely \\.\libusb0-X
4. the cleanup function

# re\_add\_device

```
1  . . .  
   lea     r9, "\\Device\\libusb0"  
   . . .  
   call    cs:_snwprintf  
   lea     r9, "\\DosDevices\\libusb0-"  
   lea     r8, "%s%04d"  
   . . .  
   call    cs:_snwprintf  
2  call    cs:IoCreateDevice  
   . . .
```

- 1. prepares the links
- 2. creates the device



# re\_major\_func

## REMINDER

- we're looking for an unknown function code `0x222048`
- the buffer size has to be `0x18`

1

```
...  
mov     eax, edi  
sub     eax, 222048h  
jz      loc_re_handle_get_version  
...
```

1. checks the IOCTL code

# re\_major\_func

## REMINDER

- we're looking for an unknown function code `0x222048`
- the buffer size has to be `0x18`

```
1  ...  
   mov     eax, edi  
   sub     eax, 222048h  
   jz      loc_re_handle_get_version  
   ...
```

```
2  ...  
   cmp     r12d, 18h  
   jb      short loc_re_error  
   ...  
3  mov     dword ptr [rsi+4], 1  
   mov     dword ptr [rsi+8], 2  
   mov     dword ptr [rsi+0Ch], 6  
   mov     [rsi+10h], r8d      ; r8d = 0  
   mov     dword ptr [rsi+14h], 1  
   ...
```

1. checks the IOCTL code
2. checks the buffer size
3. writes the value `1.2.6.0.1` inside the buffer

# USB Traffic

- The VID `0x2CA3` corresponds to 'DJI Technology Co., Ltd.'
- USB traffic can be monitored via Wireshark (USBPcap)

...	26.817005	host	1.7.0	USB	36	GET_DESCRIPTOR Request	DEVICE
27	26.817272	1.7.0	host	USB	46	GET_DESCRIPTOR Response	DEVICE
28							
...							

# USB Traffic

- The VID `0x2CA3` corresponds to 'DJI Technology Co., Ltd.'
- USB traffic can be monitored via Wireshark (USBPcap)

27	26.817005	host	1.7.0	USB	36	GET_DESCRIPTOR	Request	DEVICE
28	26.817272	1.7.0	host	USB	46	GET_DESCRIPTOR	Response	DEVICE
...								

↓

0000	1c	00	10	40	31	b5	06	cb	ff	ff	00	00	00	00	08	00	...@1.....
0010	01	01	00	07	00	80	02	12	00	00	00	03	12	01	00	02	.....
0020	00	00	00	40	a3	2c	19	00	10	03	01	02	03	01			...@.....

# USB Traffic

- The VID `0x2CA3` corresponds to 'DJI Technology Co., Ltd.'
- USB traffic can be monitored via Wireshark (USBPcap)

```
27 26.817005 host 1.7.0 USB 36 GET_DESCRIPTOR Request DEVICE
28 26.817272 1.7.0 host USB 46 GET_DESCRIPTOR Response DEVICE
...
```

```
0000 1c 00 10 40 31 b5 06 cb ff ff 00 00 00 00 08 00 ...@1.....
0010 01 01 00 07 00 80 02 12 00 00 00 03 12 01 00 02 ... ..
0020 00 00 00 40 a3 2c 19 00 10 03 01 02 03 01 ...@.....
```

```
DEVICE DESCRIPTOR
bLength: 18
bDescriptorType: 0x01 (DEVICE)
bcdUSB: 0x0200
bDeviceClass: Device (0x00)
bDeviceSubClass: 0
bDeviceProtocol: 0 (Use class code info from Interface Descriptors)
bMaxPacketSize0: 64
idVendor: DJI Technology Co., Ltd. (0x2ca3)
idProduct: Unknown (0x0019)
bcdDevice: 0x0310
iManufacturer: 1
iProduct: 2
iSerialNumber: 3
bNumConfigurations: 1
```

## DJIService.exe (IAT)

Address	Name
0x00EE1DA4	QNetworkRequest::url
0x00EE1DF4	QNetworkAccessManager::get
0x00EE1DF8	QNetworkAccessManager::post
0x00EE1E08	QNetworkReply::request
0x00EE1DD0	QSslCertificate::QSslCertificate
...	...

It uses the QT5 framework to do the job

# DJIService.exe (Network)

```
1  .:.
   push    0FFFFFFFFh
   push    offset "9dc0228943f68fb065e64b6370dbd3d966117c8..."
   lea     ecx, [ebp+var_68]
   mov     byte ptr [ebp+var_4], 13h
   call    ds:__imp_QByteArray::QByteArray
2  <A LOT OF XFORMS>
   .:.
   push    0
   push    eax
   lea     ecx, [ebp+var_90]
   mov     byte ptr [ebp+var_4], 0Fh
3  call    ds:__imp_QSslCertificate::QSslCertificate
   .:.
```

1. it uses a hex string to store the certificate data
2. a lot of decoding is done to probably make it harder to sniff
3. since we're analyzing on lowest level, we can just skip it

# DJIService.exe (Network)

```
1  .:
   push    0FFFFFFFFh
   push    offset "9dc0228943f68fb065e64b6370dbd3d966117c8..."
   lea     ecx, [ebp+var_68]
   mov     byte ptr [ebp+var_4], 13h
   call    ds:__imp_QByteArray::QByteArray
2  <A LOT OF XFORMS>
   .:
   push    0
   push    eax
   lea     ecx, [ebp+var_90]
   .:
   mov     byte ptr [ebp+var_4], 0Fh
3  call    ds:__imp_QSslCertificate::QSslCertificate
   .:
```


- 1. it uses a hex string to store the certificate data
- 2. a lot of decoding is done to probably make it harder to sniff
- 3. since we're analyzing on lowest level, we can just skip it
  - .. look inside **EAX**, as it should point to the byte array



# DJIService.exe (Network)

```
1  .:.
   push    0FFFFFFFh
   push    offset "9dc0228943f68fb065e64b6370dbd3d966117c8..."
   lea     ecx, [ebp+var_68]
   mov     byte ptr [ebp+var_4], 13h
   call    ds:__imp_QByteArray::QByteArray
2  <A LOT OF XFORMS>
   .:.
   push    0
   push    eax
   lea     ecx, [ebp+var_90]
   . mov     byte ptr [ebp+var_4], 0Fh
3  call    ds:__imp_QSslCertificate::QSslCertificate
   .:.
   .
```

- 1. it uses a hex string to store the certificate data
  - 2. a lot of decoding is done to probably make it harder to sniff
  - 3. since we're analyzing on lowest level, we can just skip it
- .. look inside **EAX**, as it should point to the byte array



015078B0	20	20	20	20	20	42	45	47	49	4E	20	50	55	42	4C	49	-----BEGIN PUBLI
015078C0	43	20	4B	45	59	20	20	20	20	20	40	49	49	42	49	6A	C KEY-----MIIBIj
015078D0	41	4E	42	67	6B	71	68	6B	69	47	39	77	30	42	41	51	ANBgkqhkiG9w0BAQ
015078E0	45	46	41	41	4F	43	41	51	38	41	40	49	49	42	43	67	EFAA0CA08AMIIBCg

# DJIService.exe (Network)

```
1  . . .
   push    0FFFFFFFh
   push    offset "9dc0228943f68fb065e64b6370dbd3d966117c8..."
   lea     ecx, [ebp+var_68]
   mov     byte ptr [ebp+var_4], 13h
   call    ds:__imp_QByteArray::QByteArray
2  <A LOT OF XFORMS>
   . . .
   push    0
   push    eax
   lea     ecx, [ebp+var_90]
   . mov     byte ptr [ebp+var_4], 0Fh
3  call    ds:__imp_QSslCertificate::QSslCertificate
   . . .
```

- 1. it uses a hex string to store the certificate data
  - 2. a lot of decoding is done to probably make it harder to sniff
  - 3. since we're analyzing on lowest level, we can just skip it
- .. look inside **EAX**, as it should point to the byte array



015078B0	20 20 20 20 20 42 45 47 49 4E 20 50 55 42 4C 49	-----BEGIN PUBLI
015078C0	43 20 4B 45 59 20 20 20 20 20 40 49 49 42 49 6A	C KEY-----MIIBIj
015078D0	41 4E 42 67 6B 71 68 6B 69 47 39 77 30 42 41 51	ANBgkqhkiG9w0BAQ
015078E0	45 46 41 41 4F 43 41 51 38 41 40 49 49 42 43 67	EFAA0CA08AMIIBCg

# DJIService.exe (PubKey)

```
-----BEGIN PUBLIC KEY-----  
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAlnRp3lNPSsPso2P/FLCT  
FUC2T/1qZgJySHavaRfzMprUyX/hf1dnYmpZ2DmsPrZDKNscYkuwUPir4nvLrs5z  
AMI24YT3s8IeZF/1wWghDusNtGu4cGxK50vyXn55xP0A0R22HvZNLCgPb6WxD3fy  
kFjK3nIqBYR2p1YC3dA0IdLXFMqZrWaQReAJi7e3EPsC9mWbjpWSpqvTGdnZU90T  
aN4k91FbfZdPZ62Vbjb1QgaBHfw72uWc+nXaXKrDzZ6az1nFB1CbT4q5P6yz5vid  
xYskufldyay1iDbK/5CvD9CSHmKmHq56NIWcpgfgxL86igBkQL8FVHI5qQkWMNtz  
TQIDAQAB  
-----END PUBLIC KEY-----
```

# Firmware cache files

- Inside the installation folder, resides the a dynamically created **firm\_cache** folder
- It contains available firmware details
- It also can contain the firmware files

```
> dir DJIEngine\DJIData\firm_cache
<DIR>          .
<DIR>          ..
               114,784 3f80beb095d3673853b5e911755318a3.cache
1,372,518,848 8121d15926da19b49616d24643c69bb4.cache
          92,574,592 9e32434cc1925c43f0280a1e3b37f26f.cache
               34,208 a296ac80b5a5618625e3ffb85516ff00.cache
               14,814 e8074b97247666efd13352148aa53778.cache
```

# Firmware cache files

- Inside the installation folder, resides the a dynamically created **firm\_cache** folder
- It contains available firmware details
- It also can contain the firmware files

```
> dir DJIEngine\DJIData\firm_cache
<DIR>          .
<DIR>          ..
             114,784  3f80beb095d3673853b5e911755318a3.cache
1,372,518,848  8121d15926da19b49616d24643c69bb4.cache
      92,574,592  9e32434cc1925c43f0280a1e3b37f26f.cache
        34,208  a296ac80b5a5618625e3ffb85516ff00.cache
        14,814  e8074b97247666efd13352148aa53778.cache
```

```
00000000  49 40 20 48 02 00 00 00  60 C0 01 00 00 00 00 00  IM*H.....
00000010  E0 00 00 00 00 01 00 00  80 BE 01 00 60 C0 01 00  ....PRAK....
00000020  00 00 00 00 02 00 00 00  50 52 41 4B 00 00 00 00  ....
.....
```

# Firmware cache files

- Inside the installation folder, resides the a dynamically created **firm\_cache** folder
- It contains available firmware details
- It also can contain the firmware files

```
> dir DJIEngine\DJIData\firm_cache
<DIR>          .
<DIR>          ..
               114,784  3f80beb095d3673853b5e911755318a3.cache
1,372,518,848  8121d15926da19b49616d24643c69bb4.cache
           92,574,592  9e32434cc1925c43f0280a1e3b37f26f.cache
               34,208  a296ac80b5a5618625e3ffb85516ff00.cache
               14,814  e8074b97247666efd13352148aa53778.cache
```

```
00000000  49 40 2a 48 02 00 00 00 60 c0 01 00 00 00 00 00  IM*H .....
00000010  E0 00 00 00 00 01 00 00 80 BE 01 00 60 c0 01 00  .....
00000020  00 00 00 00 02 00 00 00 50 52 41 4B 00 00 00 00  .....PRAK....
.....
```

## Firmware files (USBPcap)

```
265517  1455.915538 1.16.4  host  USBCOM  305  URB_BULK in
```

```

00000 1b 00 a0 48 8b b8 06 cb ff ff 00 00 00 00 09 00      . . . H . . . . .
00010 01 01 00 10 00 84 03 16 01 00 00 55 16 05 a2 0d      . . . . . U . . .
00020 2a 8f 2b c0 00 4f 00 00 01 00 00 a0 13 00 00 49      * + 0 . . . . I
00030 4d 2a 48 02 00 00 00 a0 14 00 00 00 00 00 00 e0      M*H . . . . .
00040 00 00 00 00 01 00 00 c0 12 00 00 a0 14 00 00 00      . . . . PRAK . . .
00050 00 00 00 02 00 00 00 50 52 41 4b 00 00 00 00 00      . . . . .
. . . . .

```

## Firmware files (USBPcap)

```

...
265517  1455.915538  1.16.4  host  USBCOM  305  URB_BULK in
...

```

```
0000 1b 00 a0 48 8b b8 06 cb ff ff 00 00 00 00 09 00 . . . H . . . . .
0010 01 01 00 10 00 84 03 16 01 00 00 55 16 05 a2 0d . . . . . U . . .
0020 2a 8f 2b c0 00 4f 00 00 01 00 00 a0 13 00 00 49 * . + . . . I
0030 4d 2a 48 c2 00 00 00 a0 14 00 00 00 00 00 00 e0 M * H . . . . .
0040 00 00 00 00 01 00 00 c0 12 00 00 a0 14 00 00 00 . . . . .
0050 00 00 00 02 00 00 00 50 52 41 4b 00 00 00 00 00 . . . PRAK . . .
```



# Firmware files entropy

a296ac80b5a5618625e3ffb85516ff00.cache (34KB)



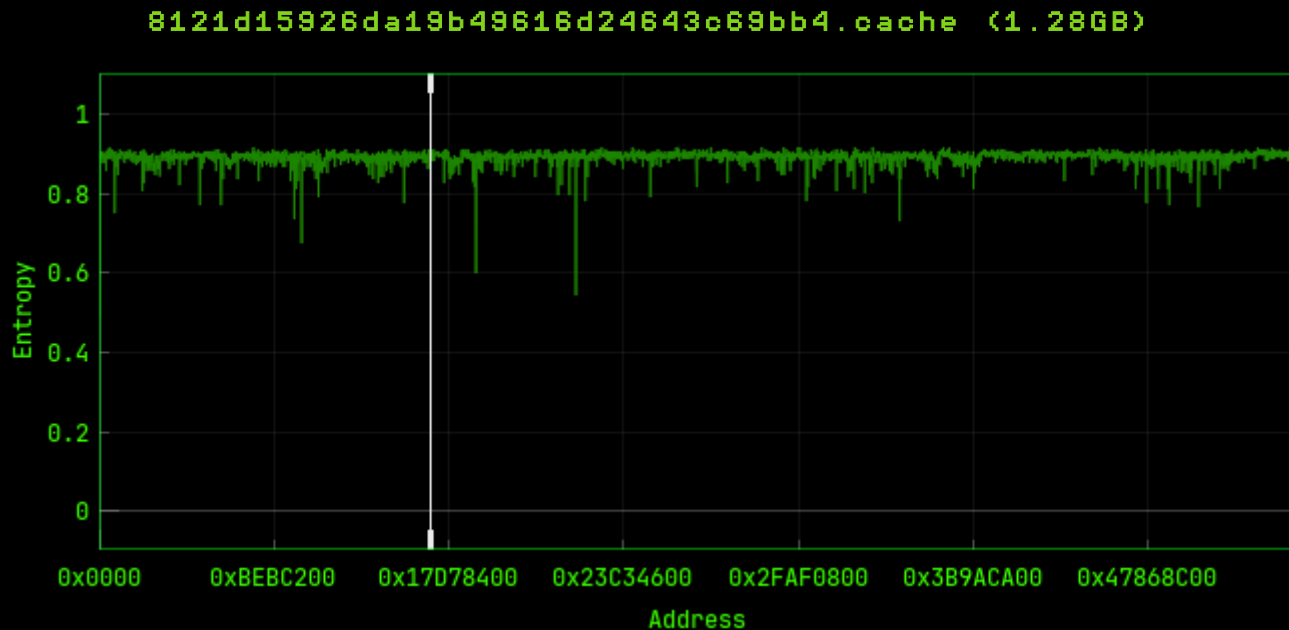
# Firmware files entropy

a296ac80b5a5618625e3ffb85516ff00.cache (34KB)



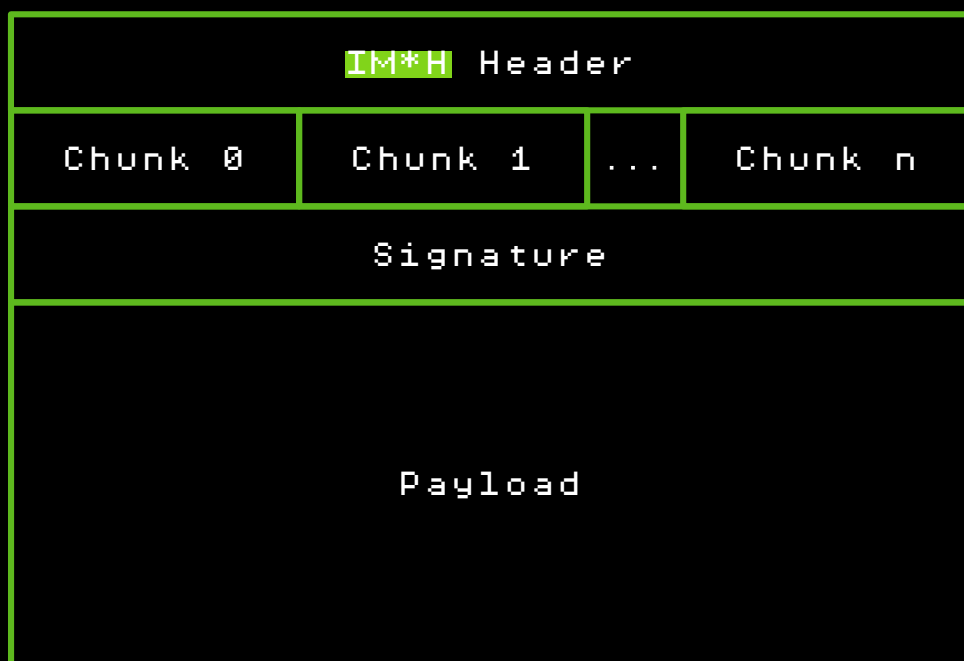
```
00002020  B1 40 6A 78 00 2A 01 D1 A1 60 00 E0 61 60 61 69  .@jx.*...`..a'ai
00002030  B0 40 08 43 60 61 70 BD 00 00 00 00 44 3A 5C 70  .@.C'ap.....D:\p
00002040  72 6F 6A 65 63 74 5C 52 4D 35 30 30 5C 65 78 74  roject\RM500\ext
00002050  5F 62 6F 61 72 74 5C 67 69 74 5C 6D 61 73 74 65  _board\git\maste
00002060  72 5C 65 78 5F 62 6F 61 72 64 5C 61 70 70 5C 68  r\ex_board\applh
...
```

# Firmware files entropy

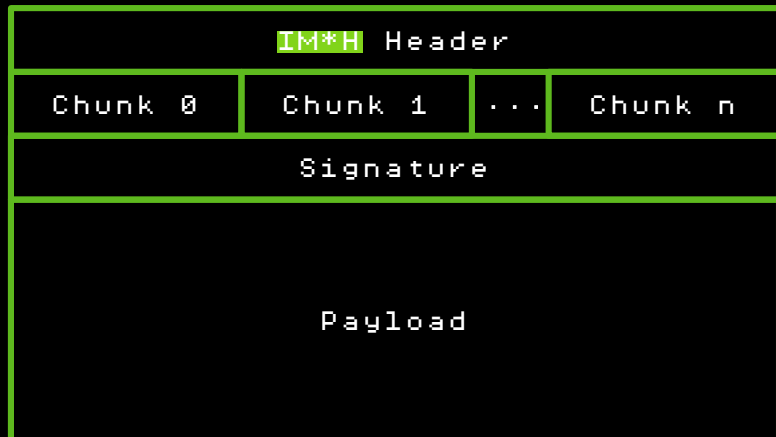


# The firmware file format

\*NOTE that this was "pre-" reverse engineered



# The firmware header



# The firmware header



```
struct imah_header {  
    s8  magic[4];  
    u32 header_version;  
    u32 size;  
    u8  reserved[4];  
    u32 header_size;  
    u32 signature_size;  
    u32 payload_size;  
    u32 target_size;  
    u8  os;  
    u8  arch;  
    u8  compression;  
    u8  anti_version;  
    u32 auth_alg;  
    u8  auth_key[4];  
    u8  enc_key[4];  
    u8  scram_key[16];  
    s8  name[32];  
    u8  type[4];  
    u32 version;  
    u32 date;  
    u32 encr_oksum;  
    u8  reserved2[16];  
    s8  userdata[16];  
    u8  entry[8];  
    u32 plain_oksum;  
    u32 chunk_num;  
    u8  payload_digest[32];  
};
```

};

# The firmware header



```
struct imah_header {  
    s8  magic[4];  
    u32 header_version;  
    u32 size;  
    u8  reserved[4];  
    u32 header_size;  
    u32 signature_size;  
    u32 payload_size;  
    u32 target_size;  
    u8  os;  
    u8  arch;  
    u8  compression;  
    u8  anti_version;  
    u32 auth_alg;  
    u8  auth_key[4];  
    u8  enc_key[4];  
    u8  scram_key[16];  
    s8  name[32];  
    u8  type[4];  
    u32 version;  
    u32 date;  
    u32 encr_oksum;  
    u8  reserved2[16];  
    s8  userdata[16];  
    u8  entry[8];  
    u32 plain_oksum;  
    u32 chunk_num;  
    u8  payload_digest[32];  
};
```

};

# The firmware header



```
struct imah_header {  
    s8  magic[4];  
    u32 header_version;  
    u32 size;  
    u8  reserved[4];  
    u32 header_size;  
    u32 signature_size;  
    u32 payload_size;  
    u32 target_size;  
    u8  os;  
    u8  arch;  
    u8  compression;  
    u8  anti_version;  
    u32 auth_alg;  
    u8  auth_key[4];  
    u8  enc_key[4];  
    u8  scram_key[16];  
    s8  name[32];  
    u8  type[4];  
    u32 version;  
    u32 date;  
    u32 encr_oksum;  
    u8  reserved2[16];  
    s8  userdata[16];  
    u8  entry[8];  
    u32 plain_oksum;  
    u32 chunk_num;  
    u8  payload_digest[32];  
};
```

};



# The firmware header



```
struct imah_header {  
    s8  magic[4];  
    u32 header_version;  
    u32 size;  
    u8  reserved[4];  
    u32 header_size;  
    u32 signature_size;  
    u32 payload_size;  
    u32 target_size;  
    u8  os;  
    u8  arch;  
    u8  compression;  
    u8  anti_version;  
    u32 auth_alg;  
    u8  auth_key[4];  
    u8  enc_key[4];  
    u8  scram_key[16];  
    s8  name[32];  
    u8  type[4];  
    u32 version;  
    u32 date;  
    u32 encr_oksum;  
    u8  reserved2[16];  
    s8  userdata[16];  
    u8  entry[8];  
    u32 plain_oksum;  
    u32 chunk_num;  
    u8  payload_digest[32];  
};
```

};

# The firmware header



```
struct imah_header {  
    s8  magic[4];  
    u32 header_version;  
    u32 size;  
    u8  reserved[4];  
    u32 header_size;  
    u32 signature_size;  
    u32 payload_size;  
    u32 target_size;  
    u8  os;  
    u8  arch;  
    u8  compression;  
    u8  anti_version;  
    u32 auth_alg;  
    u8  auth_key[4];  
    u8  enc_key[4];  
    u8  scram_key[16];  
    s8  name[32];  
    u8  type[4];  
    u32 version;  
    u32 date;  
    u32 encr_oksum;  
    u8  reserved2[16];  
    s8  userdata[16];  
    u8  entry[8];  
    u32 plain_oksum;  
    u32 chunk_num;  
    u8  payload_digest[32];  
};
```

};

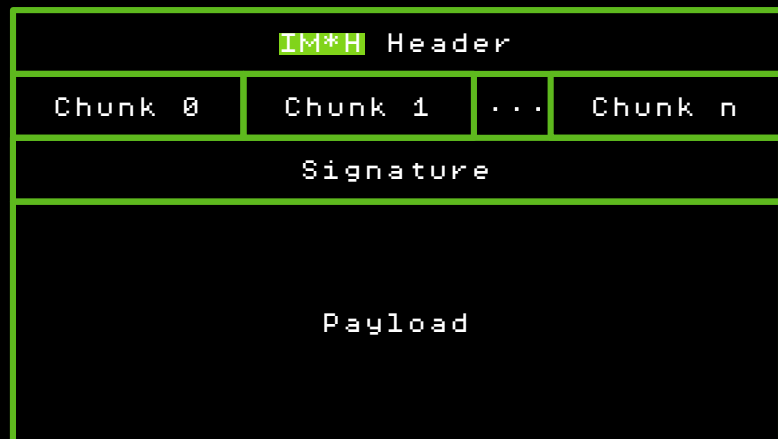
# The firmware header



```
struct imah_header {  
    s8  magic[4];  
    u32 header_version;  
    u32 size;  
    u8  reserved[4];  
    u32 header_size;  
    u32 signature_size;  
    u32 payload_size;  
    u32 target_size;  
    u8  os;  
    u8  arch;  
    u8  compression;  
    u8  anti_version;  
    u32 auth_alg;  
    u8  auth_key[4];  
    u8  enc_key[4];  
    u8  scram_key[16];  
    s8  name[32];  
    u8  type[4];  
    u32 version;  
    u32 date;  
    u32 encr_oksum;  
    u8  reserved2[16];  
    s8  userdata[16];  
    u8  entry[8];  
    u32 plain_oksum;  
    u32 chunk_num;  
    u8  payload_digest[32];  
};
```

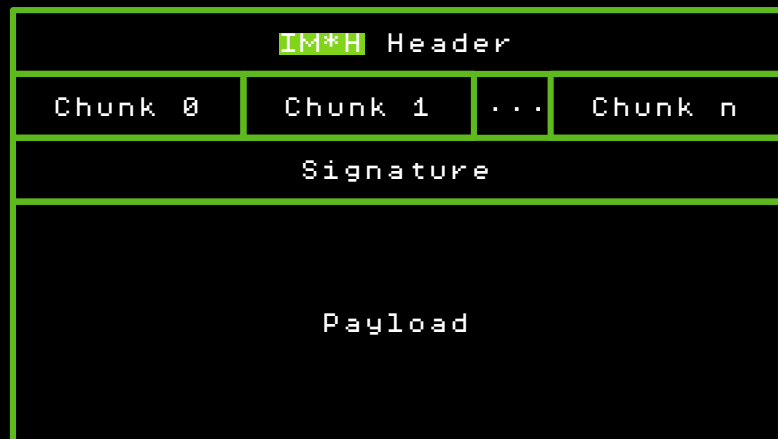
};

# The firmware chunks



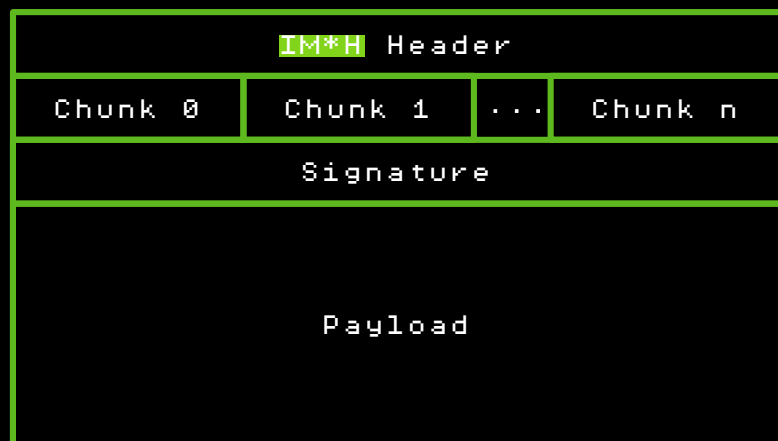
```
struct imah_chunk {  
    s8  id[4];  
    u32 offset;  
    u32 size;  
    u32 attrib;  
    u64 address;  
    u8  reserved[8];  
};
```

# The firmware chunks



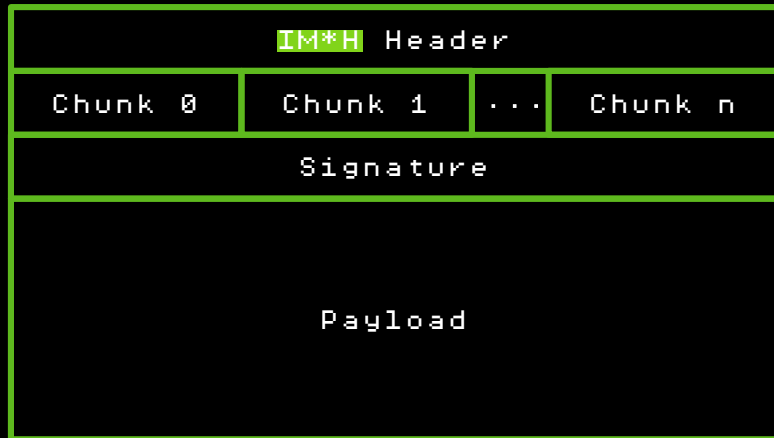
```
struct imah_chunk {  
    s8 id[4];  
    u32 offset;  
    u32 size;  
    u32 attrib;  
    u64 address;  
    u8 reserved[8];  
};
```

# The firmware signature



`u8 imah_sig[imah_header.signature_size];`

# The firmware payload



`u8 payload[];`

# Unpacking the firmware

1. Preliminary sanity checks
2. Checksum computation
3. Retrieving the encryption keys from the header hints\*
4. Read all chunk headers
5. Find the chunks, inside the payload
6. Decrypt them using the found keys\*\*



# Unpacking the firmware

1. Preliminary sanity checks
2. Checksum computation
3. Retrieving the encryption keys from the header hints\*
4. Read all chunk headers
5. Find the chunks, inside the payload
6. Decrypt them using the found keys\*\*

\* Actual keys are inside the hardware

# Unpacking the firmware

1. Preliminary sanity checks
2. Checksum computation
3. Retrieving the encryption keys from the header hints\*
4. Read all chunk headers
5. Find the chunks, inside the payload
6. Decrypt them using the found keys\*\*

\* Actual keys are inside the hardware

\*\* Uses AES to decrypt the payload

# Unpacking the firmware

1. Preliminary sanity checks
2. Checksum computation
3. Retrieving the encryption keys from the header hints\*
4. Read all chunk headers
5. Find the chunks, inside the payload
6. Decrypt them using the found keys\*\*

\* Actual keys are inside the hardware

\*\* Uses AES to decrypt the payload

GENERALLY, using `the python script` below is sufficient

# The final firmware files

```
> dir
```

```
...
```

```
33,701 a296ac80b5a5618625e3ffb85516ff00_0601.bin
```

```
118 a296ac80b5a5618625e3ffb85516ff00_0601.ini
```

```
310 a296ac80b5a5618625e3ffb85516ff00_head.ini
```

# The final firmware files

```
> dir
```

```
...
```

```
33,701 a296ac80b5a5618625e3ffb85516ff00_0601.bin
```

```
118 a296ac80b5a5618625e3ffb85516ff00_0601.ini
```

```
310 a296ac80b5a5618625e3ffb85516ff00_head.ini
```

```
...
Code:0000D000 SP_main          DCD 0x20001800
Code:0000D004                DCD unk_E02D          ; Reset
Code:0000D008                DCD NMI_handler+1      ; NMI
Code:0000D00C                DCD HardFault_handler+1 ; HardFault
...
Code:0000D038                DCD 0x92AB           ; PendSV
Code:0000D03C                DCD 0xA53B           ; SysTick
Code:0000D040                DCD IRQ_0_handler+1  ; IRQ_0
Code:0000D044                DCD IRQ_1_handler+1  ; IRQ_1
Code:0000D048                DCD IRQ_2_handler+1  ; IRQ_2
...
```

# Archived firmware files

```
$ unzip -d 9e32434cc1925c43f0280a1e3b37f26f_1301.cache
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
META-INF
boot.img
file_contexts.bin
RKLoader.bin
system.new.dat
system.patch.dat
system.transfer.list
trust.img
uboot.img
```

# The firmware filesystem

```
$ unzip -d 9e32434cc1925c43f0280a1e3b37f26f_1301.cache
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
META-INF
boot.img
file_contexts.bin
RKLoader.bin
system.new.dat
system.patch.dat
system.transfer.list
trust.img
uboot.img
```

# The firmware filesystem

```
$ unzip -d 9e32434cc1925c43f0280a1e3b37f26f_1301.cache
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
META-INF
boot.img
file_contexts.bin
RKLoader.bin
system.new.dat
system.patch.dat
system.transfer.list
trust.img
uboot.img
```

```
...
0
erase 6,99011,130560,263231,294400,347580,359936
new 2,0,1024
new 2,1024,2048
new 2,2048,3072
new 2,3072,4096
new 8,4096,4826,4827,5088,5089,5090,5091,5123
new 42,5123,5129,5130,5131,5132,5133, ...
new 2,6167,7191
new 2,7191,8215
...
```



# The firmware filesystem

```
$ unzip -d 9e32434cc1925c43f0280a1e3b37f26f_1301.cache
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
META-INF
boot.img
file_contexts.bin
RKLoader.bin
system.new.dat
system.patch.dat
system.transfer.list
trust.img
uboot.img
```

```
...
0
erase 6,99011,130560,263231,294400,347580,359936
new 2,0,1024
new 2,1024,2048
new 2,2048,3072
new 2,3072,4096
new 8,4096,4826,4827,5088,5089,5090,5091,5123
new 42,5123,5129,5130,5131,5132,5133, ...
new 2,6167,7191
new 2,7191,8215
...
```

# The firmware filesystem

```
$ unzip -d 9e32434cc1925c43f0280a1e3b37f26f_1301.cache
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
META-INF
boot.img
file_contexts.bin
RKLoader.bin
system.new.dat
system.patch.dat
system.transfer.list
trust.img
uboot.img
```

```
...
0
erase 6,99011,130560,263231,294400,347580,359936
new 2,0,1024
new 2,1024,2048
new 2,2048,3072
new 2,3072,4096
new 8,4096,4826,4827,5088,5089,5090,5091,5123
new 42,5123,5129,5130,5131,5132,5133, ...
new 2,6167,7191
new 2,7191,8215
...
```

# The firmware filesystem

```
$ unzip -d 9e32434cc1925c43f0280a1e3b37f26f_1301.cache
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
META-INF
boot.img
file_contexts.bin
RKLoader.bin
system.new.dat
system.patch.dat
system.transfer.list
trust.img
uboot.img
```

```
...
0
erase 6,99011,130560,263231,294400,347580,359936
new 2,0,1024
new 2,1024,2048
new 2,2048,3072
new 2,3072,4096
new 8,4096,4826,4827,5088,5089,5090,5091,5123
new 42,5123,5129,5130,5131,5132,5133, ...
new 2,6167,7191
new 2,7191,8215
...
```

# Building the filesystem

```
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
...
system.new.dat
system.patch.dat
system.transfer.list
...
```

# Building the filesystem

```
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
...
system.new.dat
system.patch.dat
system.transfer.list
...

$ ./sdat2img.py system.transfer.list system.new.dat system.ext4
```

# Building the filesystem

```
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
...
system.new.dat
system.patch.dat
system.transfer.list
...

$ ./sdat2img.py system.transfer.list system.new.dat system.ext4
$ mount -o loop ./system.ext4 /mnt
```

# Building the filesystem

```
$ ls 9e32434cc1925c43f0280a1e3b37f26f_1301
...
system.new.dat
system.patch.dat
system.transfer.list
...

$ ./sdat2img.py system.transfer.list system.new.dat system.ext4
$ mount -o loop ./system.ext4 /mnt
$ tree /mnt
...
dji_service
dji_verify
...
```

# Sample function

```
$ ls 8121d15926da19b49616d24643c69bb4_0205\system\bin\dji_verify  
dji_verify
```



# re\_dji\_image\_verify

```
$ ls 8121d15926da19b49616d24643c69bb4_0205\system\bin\dji_verify  
dji_verify
```

## re\_dji\_image\_verify:

```
...  
LDR             W3, [X24]  
MOV             W4, #0x482A4D49  
CMP             W3, W4  
B.NE            re_log_err  
...
```

# re\_dji\_image\_verify

```
$ ls 8121d15926da19b49616d24643c69bb4_0205\system\bin\dji_verify
dji_verify
```

re\_dji\_image\_verify:

```
...
LDR          W3, [X24]
MOV          W4, #0x482A4D49
CMP          W3, W4
B.NE        re_log_err
...
```

482A4D49 = "IM\*H"

# The processor



\*This picture was taken with a digital microscope. The chip is inside the smart controller.

# RK3399

## System Peripheral

Clock & Reset

PMU

...

Mailbox

## MCUs

Cortex-A72

Cortex-A53

Cortex-M0

## Connectivity

USB

PCIe

...

I2C

## Multi-Media Interface

Dual MIPI-CSI 4 Lane

eDP Lane

...

Dual Display Controller

## Multi-Media Processor

JPEG

Video

External Memory  
Interface

## Embedded Memory

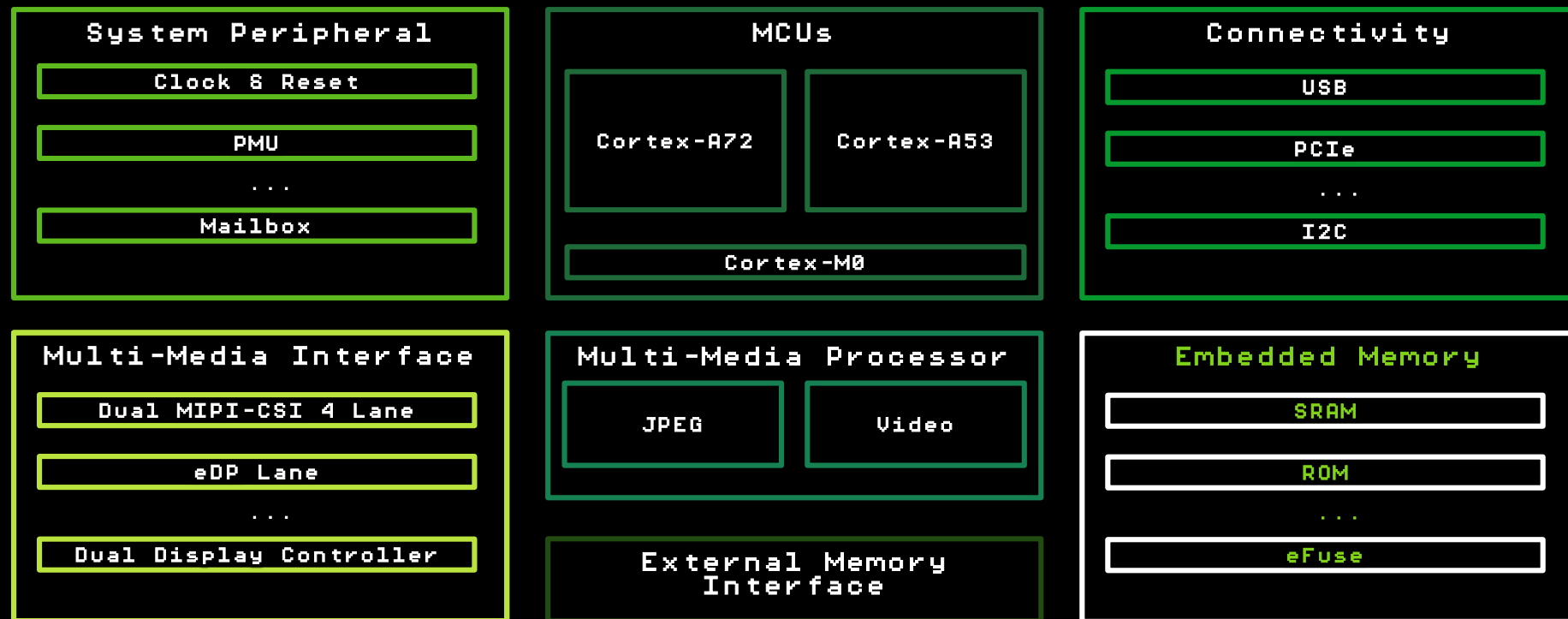
SRAM

ROM

...

eFuse

# RK3399 (Memory)



# RK3399 (Processors)

## System Peripheral

Clock & Reset

PMU

...

Mailbox

## MCUs

Cortex-A72

Cortex-A53

Cortex-M0

## Connectivity

USB

PCIe

...

I2C

## Multi-Media Interface

Dual MIPI-CSI 4 Lane

eDP Lane

...

Dual Display Controller

## Multi-Media Processor

JPEG

Video

External Memory  
Interface

## Embedded Memory

SRAM

ROM

...

eFuse

# Bootloader (bootarea.img)

```
00000000 40 00 00 00 C7 3F B7 BA 09 2E 1F EF 43 44 B9 5A @...?.....CD.Z
00000010 4E 08 05 66 B2 A3 AA AC A6 9E 3F 78 47 A7 E6 D5 N..f.....?xG...
00000020 89 6C F3 B9 F6 4E 77 1B 6C 44 F3 2E 3F AB 2E 91 .1...NW.1D...?...
00000030 AB 58 34 E4 8B BF 8E 8A D3 80 38 E8 10 AB 3D D5 .X4.....8....=
...
00000400 49 4D 2A 48 02 00 00 00 A0 5D 02 00 00 00 00 00 IM*H.....].....
00000410 00 01 00 00 00 01 00 00 A0 5B 02 00 A0 5D 02 00 .....[...l...
00000420 00 00 00 00 02 00 04 00 50 52 41 4B 54 42 49 45 .....PRAKTBIE
00000430 35 06 7C 2D 6E 38 8D 30 13 4A EA F0 29 16 12 30 5..l-n8.0.J...>...0
00000440 62 6F 6F 74 6C 6F 61 64 65 72 00 00 00 00 00 00 bootloader.....
00000450 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000460 54 42 4C 44 00 00 00 00 20 04 22 20 FA 62 F4 F4 TBLD....."..b...
00000470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000480 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000490 00 00 B8 FA 00 00 00 00 0D 18 E9 33 02 00 00 00 .....
000004A0 74 E9 56 0D F4 52 6F 02 A4 05 56 96 00 98 2B 50 t..V...Ro...U...+P
000004B0 8B EB 1B 06 94 F4 09 75 78 74 16 B5 E1 74 D9 85 .....uxt...t...
000004C0 42 4C 4C 4B 00 00 00 00 0C 5B 02 00 00 00 00 00 BLLK.....[.....
000004D0 00 00 00 FA 00 00 00 00 00 00 00 00 00 00 00 00 .....
000004E0 42 4C 46 41 20 5B 02 00 80 00 00 00 00 00 00 00 BLFA [.....
000004F0 00 E0 BA FA 00 00 00 00 00 00 00 00 00 00 00 00 .....
...
0002D000 49 4D 2A 48 02 00 00 00 00 02 00 00 00 00 00 00 IM*H.....
0002D010 E0 00 00 00 00 01 00 00 20 00 00 00 00 02 00 00 .....
0002D020 00 00 00 00 02 00 04 00 50 52 41 4B 54 42 49 45 .....PRAKTBIE
0002D030 2C B0 22 F8 7A 3C C2 00 63 89 EA 70 5B AA 76 80 ,..".z<...c..pl.v.
0002D040 62 6F 61 72 64 69 6E 66 6F 00 00 00 00 00 00 00 boardinfo.....
0002D050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0002D060 42 49 4E 46 00 00 00 00 20 04 22 20 34 02 95 65 BINF....."..4...e
0002D070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

# Bootloader (bootarea.img)

```
00000000 40 00 00 00 C7 3F B7 BA 09 2E 1F EF 43 44 B9 5A @...?.....CD.Z
00000010 4E 08 05 66 B2 A3 AA AC A6 9E 3F 78 47 A7 E6 D5 N..f.....?xG...
00000020 89 6C F3 B9 F6 4E 77 1B 6C 44 F3 2E 3F AB 2E 91 .1...NW.1D...?...
00000030 AB 58 34 E4 8B BF 8E 8A D3 80 38 E8 10 AB 3D D5 .X4.....8....=
...
00000400 49 4D 2A 48 02 00 00 00 A0 5D 02 00 00 00 00 IM*H.....].....
00000410 00 01 00 00 00 01 00 00 A0 5B 02 00 A0 5D 02 00 .....[...].
00000420 00 00 00 00 02 00 04 00 50 52 41 4B 54 42 49 45 .....PRAKTBIE
00000430 35 06 7C 2D 6E 38 8D 30 13 4A EA F0 29 16 12 30 5..l-n8.0.J...>..0
00000440 62 6F 6F 74 6C 6F 61 64 65 72 00 00 00 00 00 00 bootloader.....
00000450 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000460 54 42 4C 44 00 00 00 00 20 04 22 20 FA 62 F4 F4 TBLD....."..b..
00000470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000480 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000490 00 00 B8 FA 00 00 00 00 0D 18 E9 33 02 00 00 00 .....
000004A0 74 E9 56 0D F4 52 6F 02 A4 05 56 96 00 98 2B 50 t..V...Ro...U...+P
000004B0 8B EB 1B 06 94 F4 09 75 78 74 16 B5 E1 74 D9 85 .....uxt...t..
000004C0 42 4C 4C 4B 00 00 00 00 0C 5B 02 00 00 00 00 00 BLLK.....[.....
000004D0 00 00 00 FA 00 00 00 00 00 00 00 00 00 00 00 00 .....
000004E0 42 4C 46 41 20 5B 02 00 80 00 00 00 00 00 00 00 BLFA [.....
000004F0 00 E0 BA FA 00 00 00 00 00 00 00 00 00 00 00 00 .....
...
0002D000 49 4D 2A 48 02 00 00 00 00 02 00 00 00 00 00 IM*H.....
0002D010 F0 00 00 00 00 01 00 00 20 00 00 00 00 02 00 00 .....
0002D020 00 00 00 00 02 00 04 00 50 52 41 4B 54 42 49 45 .....PRAKTBIE
0002D030 2C B0 22 F8 7A 3C C2 00 63 89 EA 70 5B AA 76 80 ,..".z<...c..pl.v.
0002D040 62 6F 61 72 64 69 6E 66 6F 00 00 00 00 00 00 00 boardinfo.....
0002D050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0002D060 42 49 4E 46 00 00 00 00 20 04 22 20 34 02 95 65 BINF....."..4...e
0002D070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```



# Bootloader (bootarea.img)

```
00000000 40 00 00 00 C7 3F B7 BA 09 2E 1F EF 43 44 B9 5A @...?.....CD.Z
00000010 4E 08 05 66 B2 A3 AA AC A6 9E 3F 78 47 A7 E6 D5 N..f.....?xG...
00000020 89 6C F3 B9 F6 4E 77 1B 6C 44 F3 2E 3F AB 2E 91 .1...NW.1D...?...
00000030 AB 58 34 E4 8B BF 8E 8A D3 80 38 E8 10 AB 3D D5 .X4.....8....=
...
00000400 49 4D 2A 48 02 00 00 00 A0 5D 02 00 00 00 00 00 IM*H.....].....
00000410 00 01 00 00 00 01 00 00 A0 5B 02 00 A0 5D 02 00 .....[...].
00000420 00 00 00 00 02 00 04 00 50 52 41 4B 54 42 49 45 .....PRAKTBIE
00000430 35 06 7C 2D 6E 38 8D 30 13 4A EA F0 29 16 12 30 5..l-n8.0.J...0
00000440 62 6F 6F 74 6C 6F 61 64 65 72 00 00 00 00 00 00 bootloader.....
00000450 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 TBLD....."..b..
00000460 54 42 4C 44 00 00 00 00 20 04 22 20 FA 62 F4 F4 .....
00000470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000480 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000490 00 00 B8 FA 00 00 00 00 0D 18 E9 33 02 00 00 00 .....
000004A0 74 E9 56 0D F4 52 6F 02 A4 05 56 96 00 98 2B 50 t..V...Ro...U...+P
000004B0 8B EB 1B 06 94 F4 09 75 78 74 16 B5 E1 74 D9 85 .....uxt...t..
000004C0 42 4C 4C 4B 00 00 00 00 0C 5B 02 00 00 00 00 00 BLLK.....[.....
000004D0 00 00 00 FA 00 00 00 00 00 00 00 00 00 00 00 00 BLFA [.....
000004E0 42 4C 46 41 20 5B 02 00 80 00 00 00 00 00 00 00 .....
000004F0 00 E0 BA FA 00 00 00 00 00 00 00 00 00 00 00 00 .....
...
0002D000 49 4D 2A 48 02 00 00 00 00 02 00 00 00 00 00 00 IM*H.....
0002D010 E0 00 00 00 00 01 00 00 20 00 00 00 00 02 00 00 .....
0002D020 00 00 00 00 02 00 04 00 50 52 41 4B 54 42 49 45 .....PRAKTBIE
0002D030 2C B0 22 F8 7A 3C C2 08 63 89 EA 70 5B AA 76 80 ..".z<...c..pl.v.
0002D040 62 6F 61 72 64 69 6E 66 6F 00 00 00 00 00 00 00 boardinfo.....
0002D050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0002D060 42 49 4E 46 00 00 00 00 20 04 22 20 34 02 95 65 BINF....." 4..e
0002D070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

# Bootloader

ROM:00000000	B	loc_20
ROM:00000004	B	loc_1A70
ROM:00000008	B	loc_1AC0
ROM:0000000C	B	loc_1B20
ROM:00000010	B	loc_1B7C
ROM:00000014	B	loc_1BD8
ROM:00000018	B	loc_1BDC
ROM:0000001C	B	loc_1C60
...		

# Bootloader

ROM:00000000	B	loc_20
ROM:00000004	B	loc_1A70
ROM:00000008	B	loc_1AC8
ROM:0000000C	B	loc_1B20
ROM:00000010	B	loc_1B7C
ROM:00000014	B	loc_1BD8
ROM:00000018	B	loc_1BDC
ROM:0000001C	B	loc_1C60
...		

## Vector Table

0x00: Reset

0x04: Undefined

0x08: Syscall

0x0C: Prefetch Abort

0x10: Data Abort

0x14: Reserved

0x18: IRQ

0x1C: FRQ

# Bootloader

ROM:00000000	B	re_reset
ROM:00000004	B	re_undef
ROM:00000008	B	loc_1AC8
ROM:0000000C	B	loc_1B20
ROM:00000010	B	loc_1B7C
ROM:00000014	B	loc_1BD8
ROM:00000018	B	loc_1BDC
ROM:0000001C	B	loc_1C60
...		

## Vector Table

0x00: Reset

0x04: Undefined

0x08: Syscall

0x0C: Prefetch Abort

0x10: Data Abort

0x14: Reserved

0x18: IRQ

0x1C: FRQ

# Bootloader

ROM:00000000	B	re_reset
ROM:00000004	B	re_undef
ROM:00000008	B	loc_1AC8
ROM:0000000C	B	loc_1B20
ROM:00000010	B	loc_1B7C
ROM:00000014	B	loc_1BD8
ROM:00000018	B	loc_1BDC
ROM:0000001C	B	loc_1C60
...		

re\_reset:

MRC	p15, 0, R12, c1, c0, 0
BIC	R12, R12, #0x1000
BIC	R12, R12, #5
MRC	p15, 1, R0, c0, c0, 0
...	

# Bootloader

ROM:00000000	B	re_reset
ROM:00000004	B	re_undef
ROM:00000008	B	loc_1AC8
ROM:0000000C	B	loc_1B20
ROM:00000010	B	loc_1B7C
ROM:00000014	B	loc_1BD8
ROM:00000018	B	loc_1BDC
ROM:0000001C	B	loc_1C60
...		

```
re_reset:
    MRC                p15, 0, R12,c1,c0, 0
    BIC                R12, R12, #0x1000
    BIC                R12, R12, #5
    MRC                p15, 1, R0,c0,c0, 0
    ...
    BLX                re_1k_main
```

# Bootloader

```
re_1k_main:
    .L1:
    BL      re_dprintf
    LDR     R0, "Enter 1k_main\n"
    .L2:
    BL      re_thread_init_early
    .L3:
```

# LittleKernel

```
re_lk_main:
    ...
    BL      re_dprintf
    LDR     R0, "Enter lk_main\n"
    ...
    BL      re_thread_init_early
    ...
```

## main.c

```
void lk_main(ulong arg0, ulong arg1, ulong arg2, ulong arg3) {
    // save the boot args
    lk_boot_args[0] = arg0;
    lk_boot_args[1] = arg1;
    lk_boot_args[2] = arg2;
    lk_boot_args[3] = arg3;

    // get us into some sort of thread context
    thread_init_early();
    ...
}
```



## Related Work

### Paper:

- Title: Challenges in Dynamic Analysis of Drone Firmware and Its Solutions
- DOI: 10.1109/ACCESS.2024.3425604

### Paper:

- Title: Drone Security and the Mysterious Case of DJI's DroneID
- DOI: 10.14722/ndss.2023.24217

### Repository:

- Link: <https://github.com/o-gs/dji-firmware-tools>

# Questions?

ASTRONAUT:

- Website: [astronaut.am](http://astronaut.am)

Hrant Tadevosyan

- LinkedIn: <https://am.linkedin.com/in/hrant-tadevosyan-a75741200>
- Email: [hrant.tadevosyan@protonmail.com](mailto:hrant.tadevosyan@protonmail.com)

Levon Martirosyan\*

- LinkedIn: <https://www.linkedin.com/in/levon-martirosyan-b98251189>
- Email: [levonmartirosyan2019@gmail.com](mailto:levonmartirosyan2019@gmail.com)

Github: <https://github.com/0x0000z3r0/dji-re>

# BONUS: Encryption Keys

Inside the `dji_verify` or `bootarea` we can see the embedded values

```
ED 9E 47 32 93 80 01 00 01 00 40 00 00 00 C3 15
16 41 15 7D 30 44 8F EE 89 58 D6 84 33 2E 8B 28
21 3C DB 05 C9 23 E0 6A FE 2D 13 37 1B 48 87 C2
87 2F 7F D6 74 49 0E 25 00 17 18 3A 9F CF B4 10
9F DD D8 6A 55 5F C8 74 B0 8D 64 19 C4 B7 FA 7E
03 B8 F1 06 A0 8F 57 1E 8C 26 A5 32 FC 23 E1 DD
```

```
"PRAK-2017-12": bytes.fromhex((
# Provisioning RSA Auth Key v3; published 2021-09-30 by Mefistotelis
# first use on 2017-12-14; used for:
# RC230 FW V01.00.0000,
# RC240 FW V01.00.0640,
# WM240 FW V00.06.0000-V01.00.0670 inside m090?,
# WM245 FW V01.01.0000-V01.01.0800 inside m090?,
# WM246 FW V01.00.0000-V01.01.0800 inside m090?,
"40000000c3151641157d30448fee8958d684332e8b28213cdb05c923e06afe2d"
"13371b4887c2872f7fd674490e250017183a9fcfb4109fddd86a555fc874b08d"
"6419c4b7fa7e03b8f106a08f571e8c26a532fc23e1dd0d7fe4d496523b08bc50"
"d9238a6baab57d37a13f3afd91284c8b98e2b45ecb87bdcbe691d8764f907729"
"0b236c0d8df4b2eb3ba2f36671967aaefb4ec263c9e4d75006d97f60a5eb8848"
"4b42707d0a28b9a116526acf8bc98e7e97aaa09aa5e2c8b6aaab7a2c21283c73")
```

This slide is for careful readers

## BONUS: Protocol

Its called **DUML**. WireShark dissectors do exist

Source: [https://github.com/o-gs/dji-firmware-tools/tree/master/comm\\_dissector/wireshark](https://github.com/o-gs/dji-firmware-tools/tree/master/comm_dissector/wireshark)

## BONUS: Chunk IDs

They correspond to 'hardware' modules