CIS 700/002 : Special Topics : Metaphor – a real-life Stagefright exploit

Hung Nguyen

CIS 700/002: Security of EMBS/CPS/IoT

Department of Computer and Information Science

School of Engineering and Applied Science

University of Pennsylvania

02/17/2017

What is Stagefright

- One of the most notorious Android's vulnerabilities
  - User doesn't have to do anything to accept the bug
- Exploits Android core library *libStageFright*
  - Available since Android 2.2
  - Media playback engine for popular media formats
- "Android devices with a security patch level of October 1, 2015 or greater are protected" - Google

Impractical to Exploit In-The-Wild

- Enforced execute protections on memory
  - Non-executable memory
  - Code signing

- Address Space Layout Randomization (ASLR)
  - Available since Android 4.0
  - Randomly arranges the address space positions of key data areas (executable base, libraries, etc.)

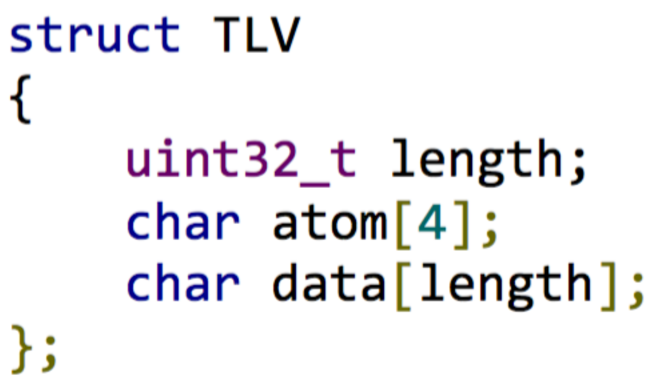
Metaphor

- Exploits Stagefright with more generic and practical approach
  - Practical = fast, reliable, stealthy

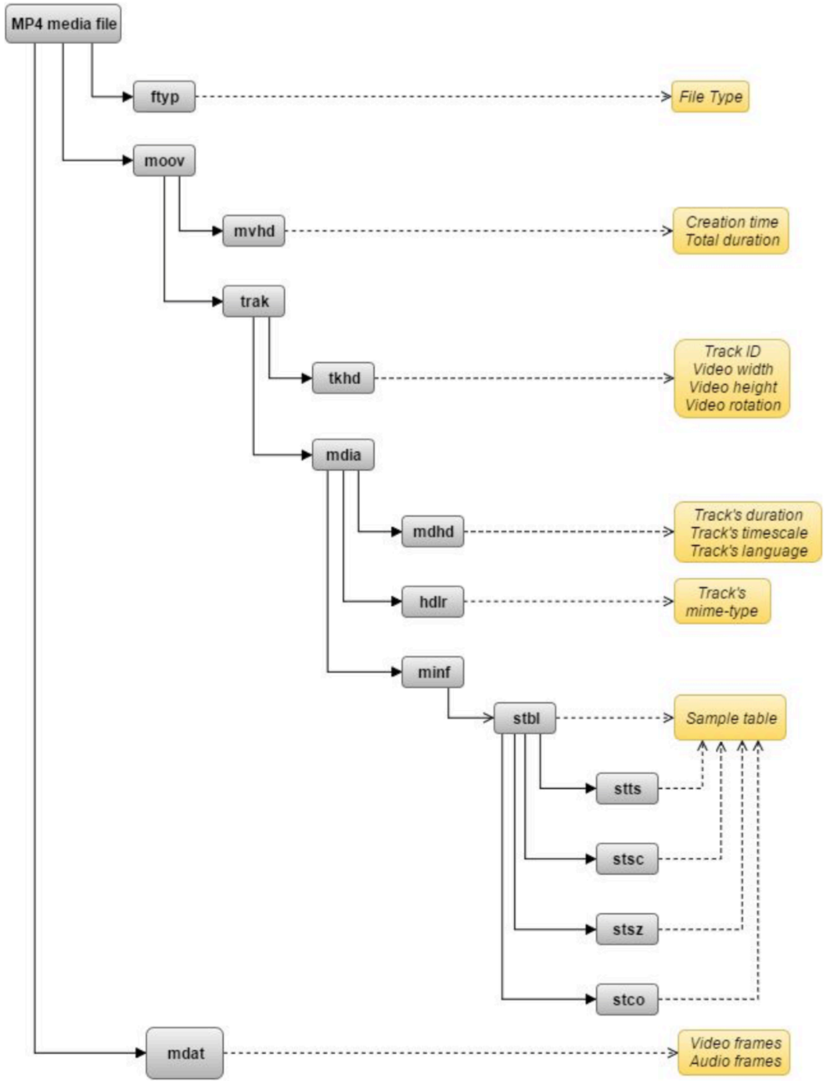
- Bypass ASLR

MPEG-4 File Format

- A collection of TLV (Type--Length--Value) chunks



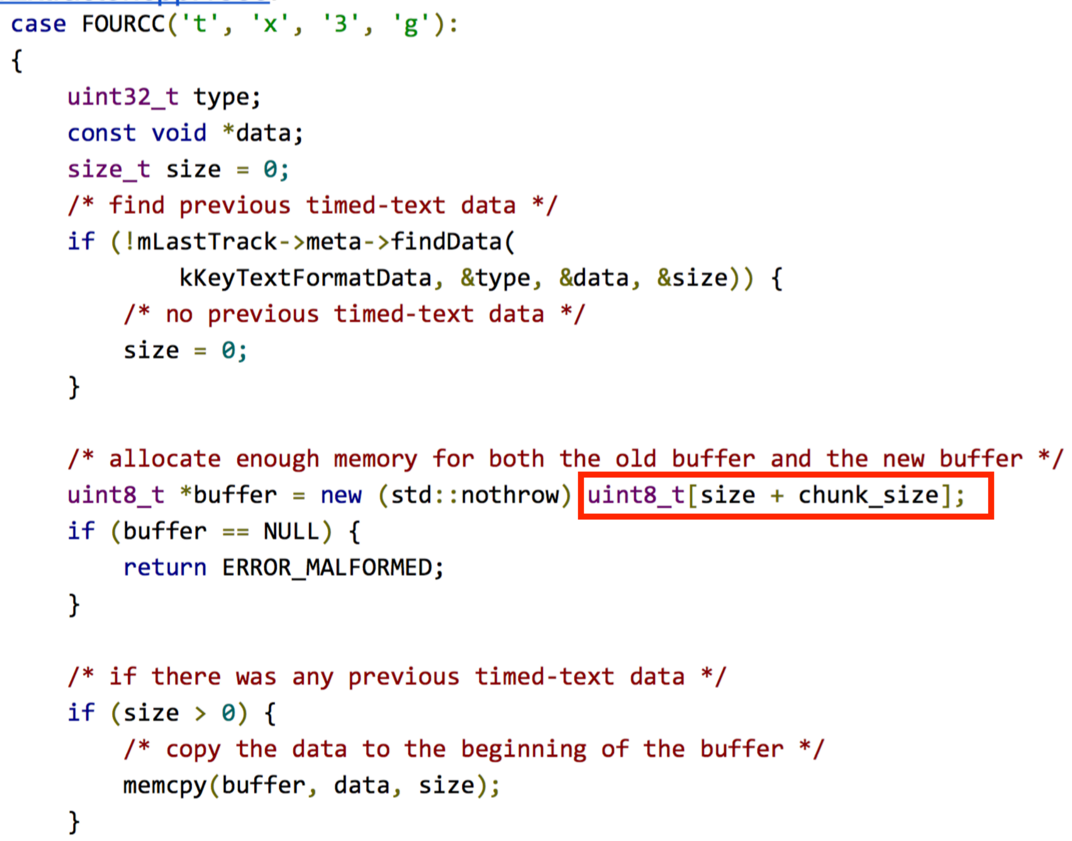
```
struct TLV
{
     uint32_t length;
     char atom[4];
     char data[length];
};
```



The Bug – CVE-2015-3864

- size & chunk_size are unchecked and allowing to cause an integer overflow

MPEG4Extractor.cpp:1886:



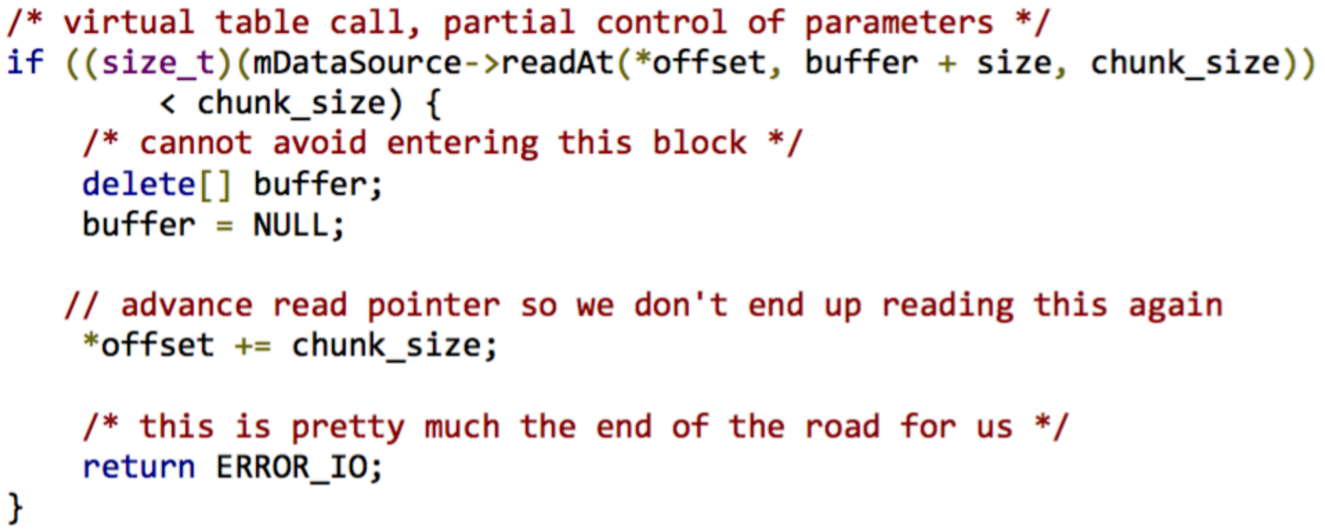
```
case FOURCC('t', 'x', '3', 'g'):
{
    uint32_t type;
    const void *data;
    size_t size = 0;
    /* find previous timed-text data */
    if (!mLastTrack->meta->findData(
            kKeyTextFormatData, &type, &data, &size)) {
        /* no previous timed-text data */
        size = 0;
    }

    /* allocate enough memory for both the old buffer and the new buffer */
    uint8_t *buffer = new (std::nothrow) uint8_t[size + chunk_size];
    if (buffer == NULL) {
        return ERROR_MALFORMED;
    }

    /* if there was any previous timed-text data */
    if (size > 0) {
        /* copy the data to the beginning of the buffer */
        memcpy(buffer, data, size);
    }
```

The Bug – CVE-2015-3864

- Shapes the heap so that the mDataSource is allocated right after the overflowed buffer
- Overwrites mDataSource’s virtual table to our own and set the respective readAt entry to point to our own memory (CVE-2015-3864)



```
/* virtual table call, partial control of parameters */
if ((size_t)(mDataSource->readAt(*offset, buffer + size, chunk_size))
        < chunk_size) {
    /* cannot avoid entering this block */
    delete[] buffer;
    buffer = NULL;

   // advance read pointer so we don't end up reading this again
    *offset += chunk_size;

    /* this is pretty much the end of the road for us */
    return ERROR_IO;
}
```

What Next?

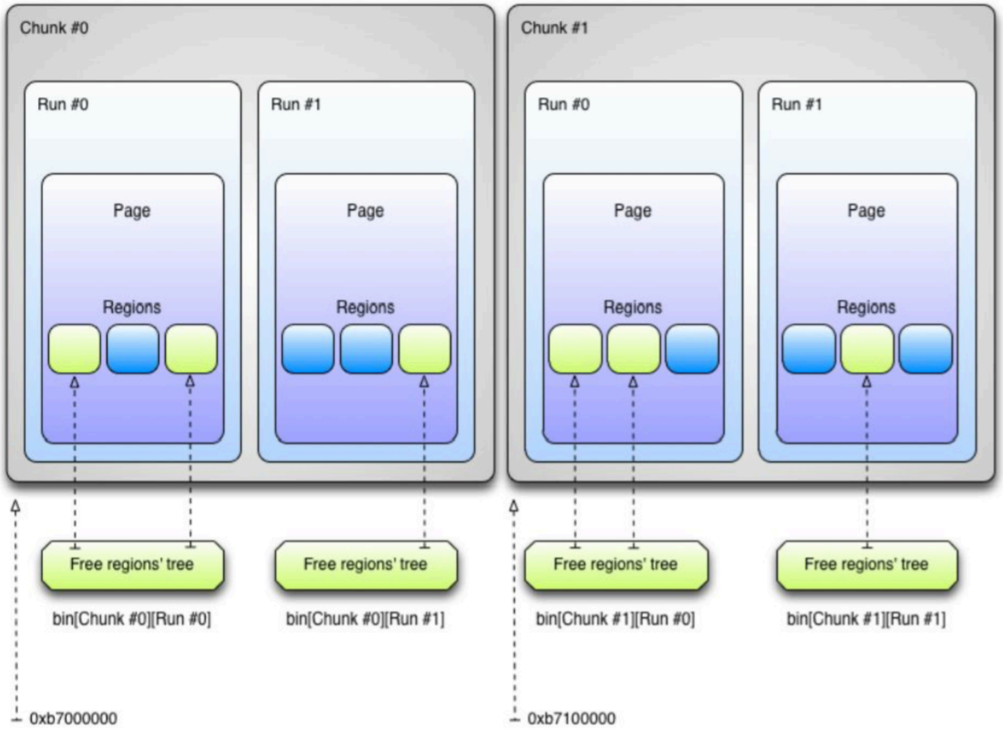
- Gives us full control of the virtual table
  - Redirecting any method to any code address

- Requires knowing or guessing our fake table's address
  - Predictable as shown by Google Project Zero: Stagefrightened

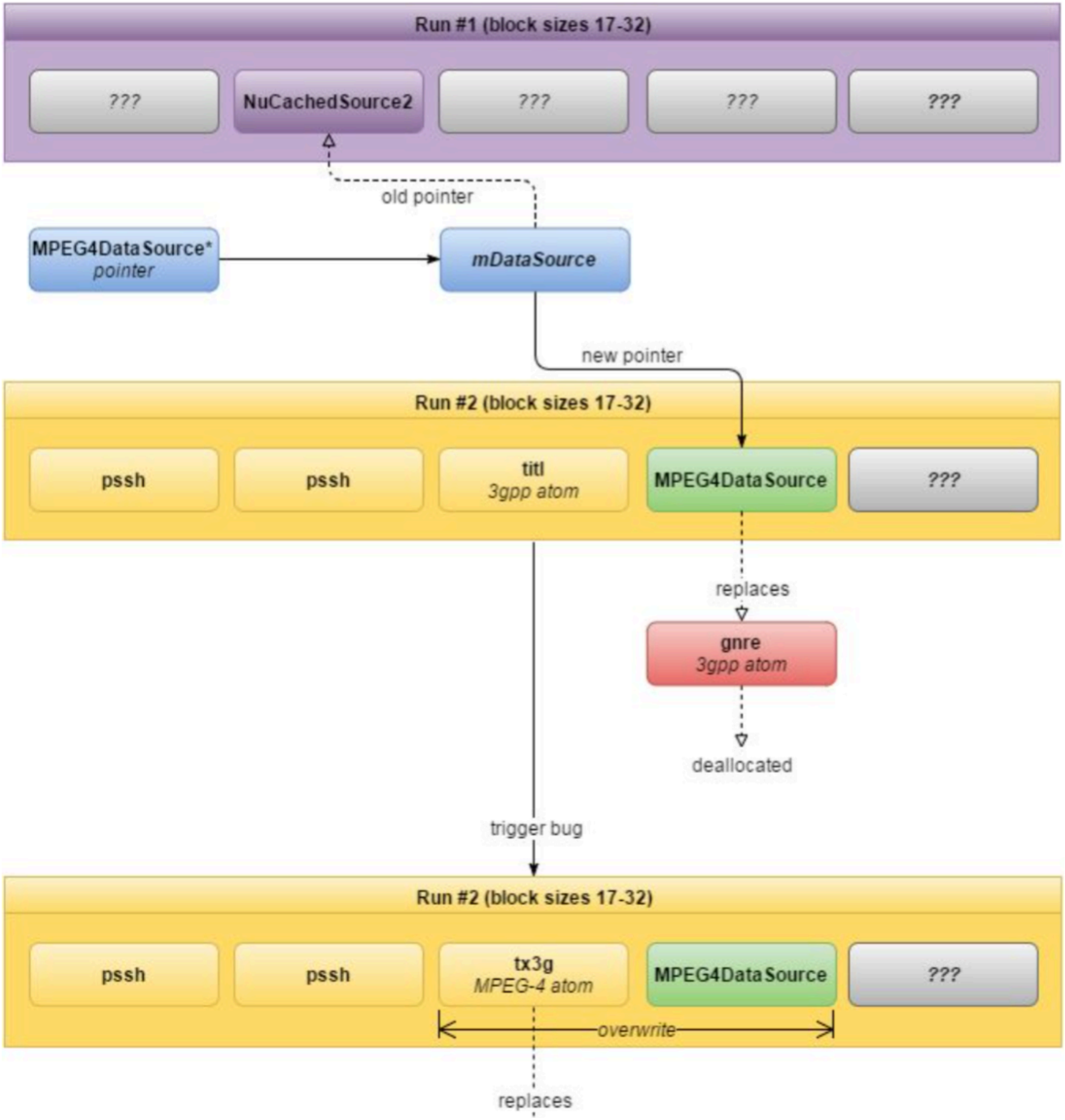
- Requires knowing libc.so function addresses for ROP chain gadgets
  - i.e. breaking ASLR!

Android Heap Allocator - jemalloc

- Allocates objects of similar sizes in the same run
  - A run is basically an array of buffers of the same size called regions
  - Objects sizes slightly smaller than the respective region's fixed size will be rounded up.

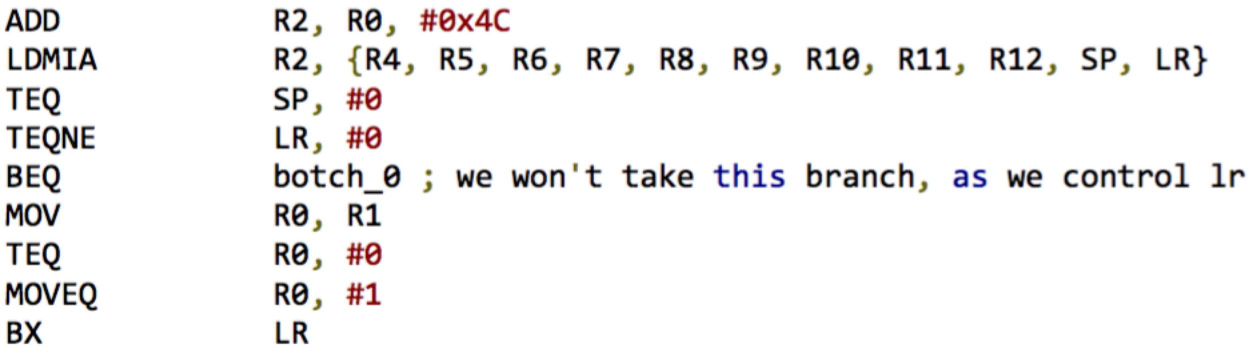


- Heap spraying
- Heap grooming


Run #1 (block sizes 17-32)
???
NuCachedSource2
???
???
???
old pointer
MPEG4DataSource*
pointer
mDataSource
new pointer
Run #2 (block sizes 17-32)
pssh
pssh
titl
3gpp atom
MPEG4DataSource
???
replaces
gnre
3gpp atom
deallocated
trigger bug
Run #2 (block sizes 17-32)
pssh
pssh
tx3g
MPEG-4 atom
MPEG4DataSource
???
overwrite
replaces

ROP Chain Gadgets

- Allows executing code in the presence of non-executable memory or code signing
  - hijacks program control flow
  - executes carefully chosen machine instruction sequences that are already in machine's memory
- Chains gadgets to copy in shellcode and jump to it using only functions from within libc.so



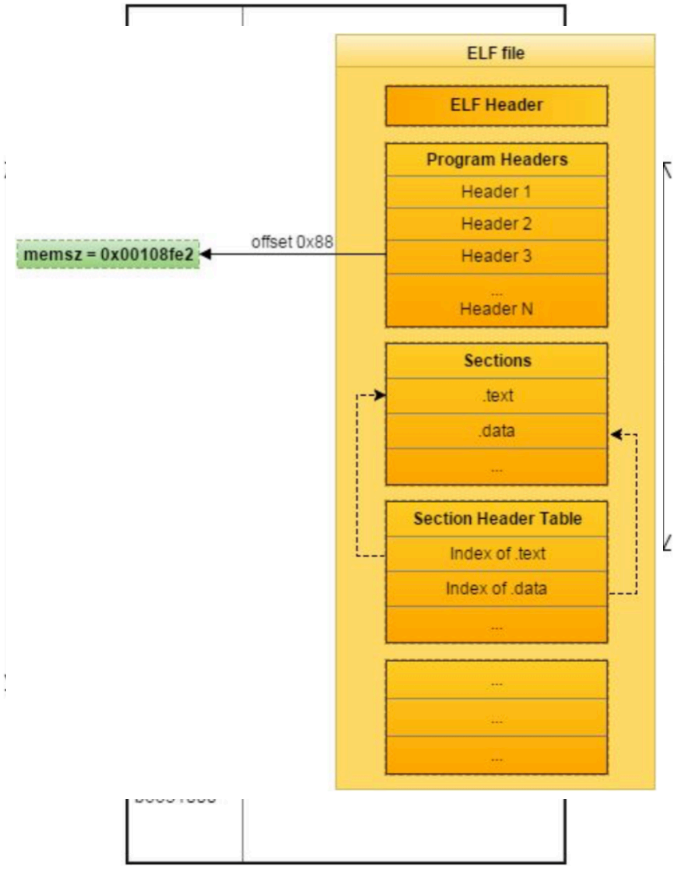
```
ADD         R2, R0, #0x4C
LDMIA       R2, {R4, R5, R6, R7, R8, R9, R10, R11, R12, SP, LR}
TEQ         SP, #0
TEQNE       LR, #0
BEQ         botch_0 ; we won't take this branch, as we control lr
MOV         R0, R1
TEQ         R0, #0
MOVEQ       R0, #1
BX          LR
```

ASLR Weakness

- ASLR on 32-bit ARM simply moves all modules to a random amount of pages down (0-255)
  - *ASLR slide* is only generated on process startup

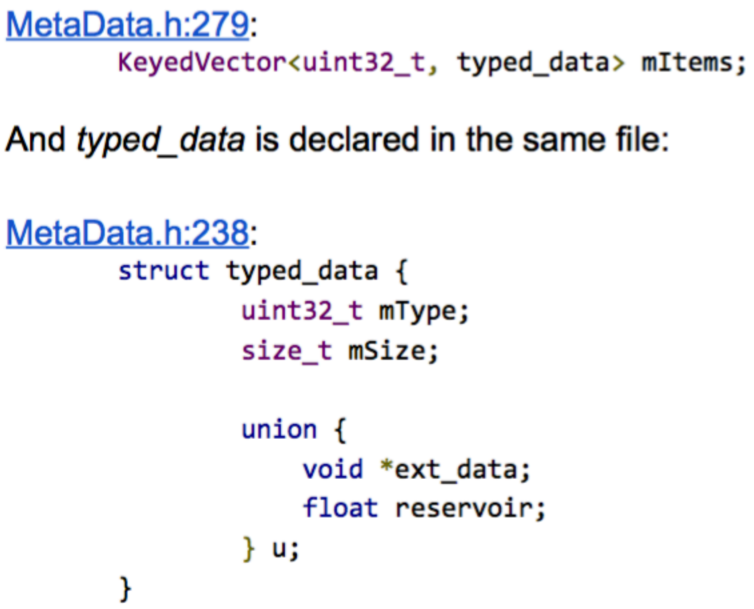
- p_memsz
  - unique to each module
  - fixed offset 0x88
  - readable

→ used to detect ASLR slide



Leaking Information

- Metadata is stored in *MetaData* objects
  - multiple *mItems* fields



MetaData.h:279:

```
        KeyedVector<uint32_t, typed_data> mItems;
```

And *typed_data* is declared in the same file:

MetaData.h:238:

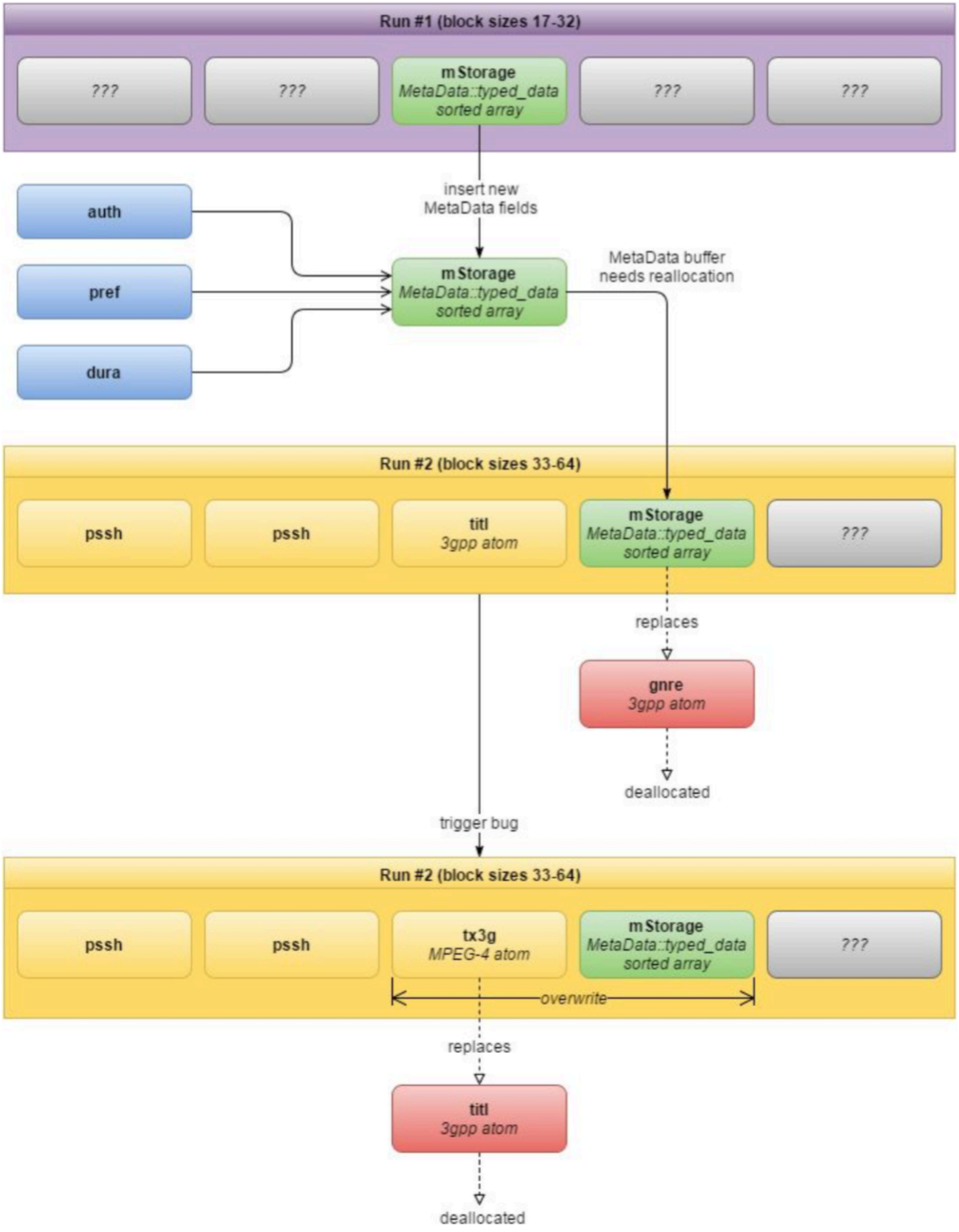
```
        struct typed_data {
                uint32_t mType;
                size_t mSize;

                union {
                    void *ext_data;
                    float reservoir;
                } u;
        }
```

- If mSize > 4

ext_data will point to memory where the data is held

Memory leak is achieved through *duration* field.


Run #1 (block sizes 17-32)
???
???
mStorage
MetaData::typed_data
sorted array
???
???
insert new
MetaData fields
auth
pref
dura
mStorage
MetaData::typed_data
sorted array
MetaData buffer
needs reallocation
Run #2 (block sizes 33-64)
pssh
pssh
titl
3gpp atom
mStorage
MetaData::typed_data
sorted array
???
replaces
gnre
3gpp atom
deallocated
trigger bug
Run #2 (block sizes 33-64)
pssh
pssh
tx3g
MPEG-4 atom
mStorage
MetaData::typed_data
sorted array
???
overwrite
replaces
titl
3gpp atom
deallocated

Breaking ASLR

- Webpage contains JavaScript scripts
  - Access metadata inside media files (videoWidth, etc.)

  → allow arbitrary memory sent back to browser

- Victim has to download/parse up to 256 media files
  - To find ELF header → fixed gadget absolute address

- HTTP supports GZIP to compress content
  - Media file is around 32MB → gzip to 32kB

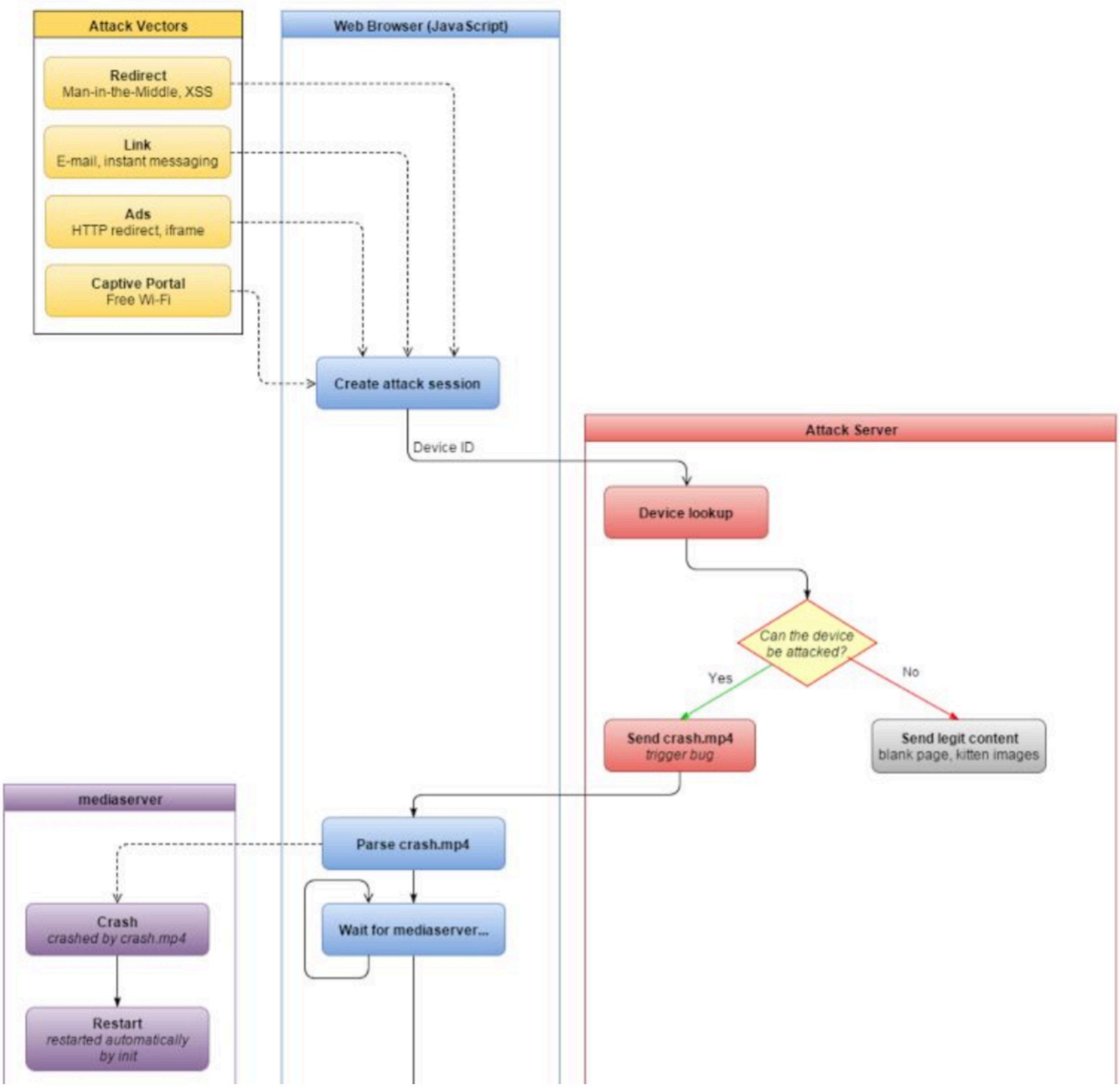
Put It All Together

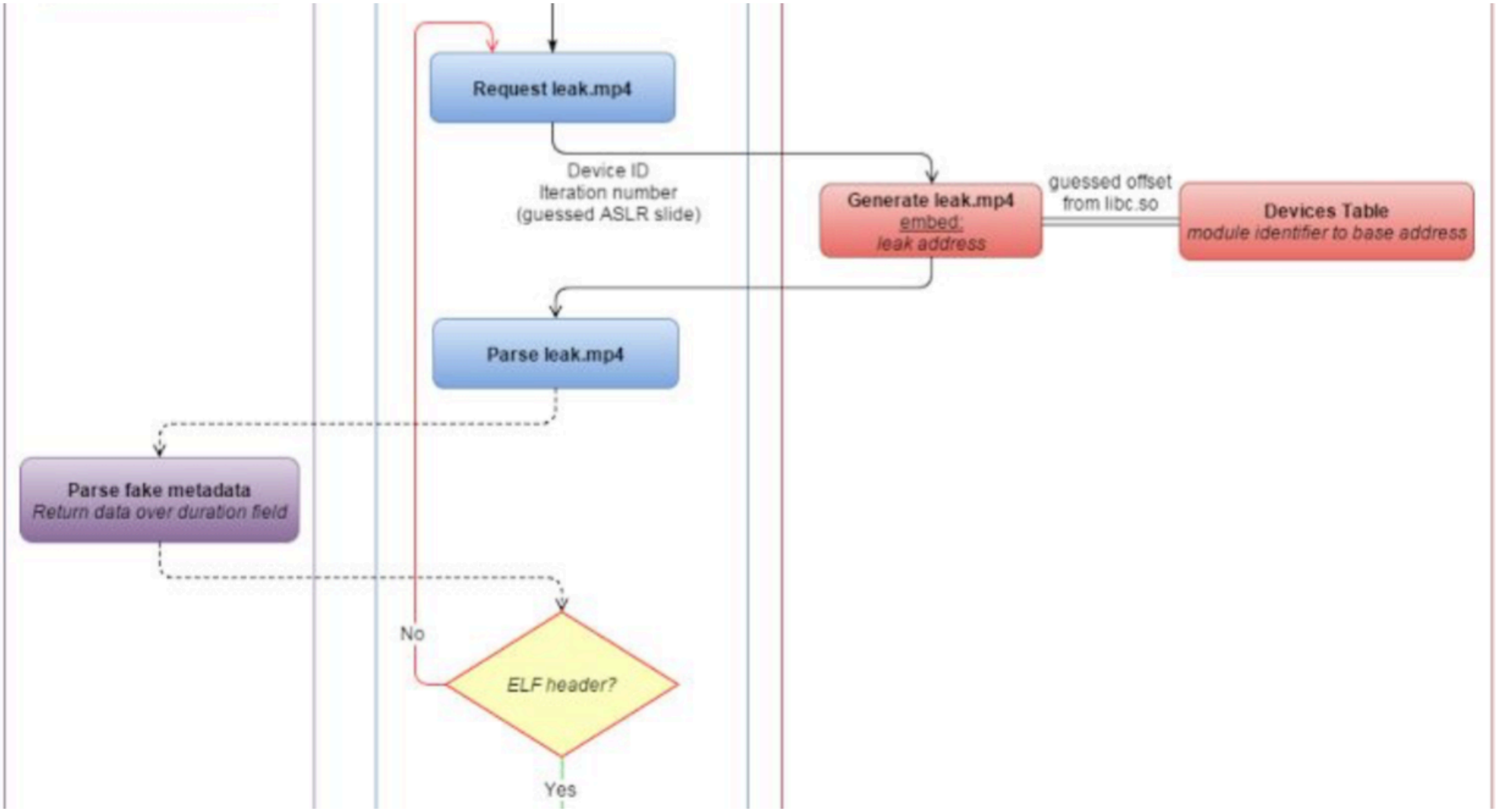
- Crash
  - Generates a small and generic media file
  - Crashes mediaserver to reset its state
- Leak
  - Generates a device--customized media file to leak memory from the mediaserver process
  - Information is returned through the *duration* field of the <video> tag
- RCE
  - Generates a device-customized media file executing shellcode in mediaserver

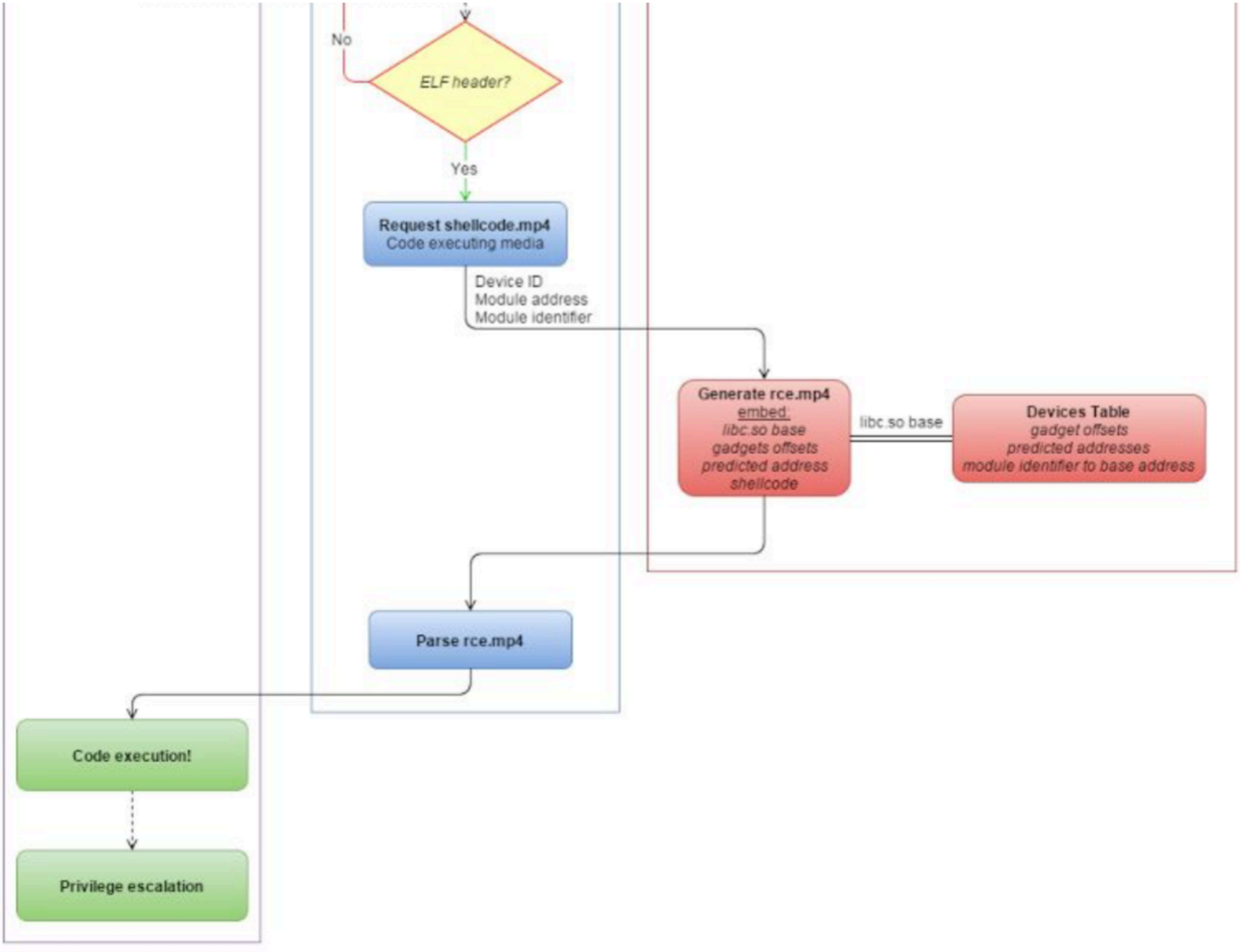
Attack Vectors

- Webpage with malicious JavaScript scripts
  - Victim only needs to parse the media file

- Different methods to lure victim to webpage
  - Ads
  - Drive-by (free Wi-Fi, QR code, etc.)
  - XSS (trusted website with malicious content)


Attack Vectors
Redirect
Man-in-the-Middle, XSS
Link
E-mail, instant messaging
Ads
HTTP redirect, iframe
Captive Portal
Free Wi-Fi
Web Browser (JavaScript)
Create attack session
Device ID
Attack Server
Device lookup
Can the device be attacked?
Yes
No
Send crash.mp4
trigger bug
Send legit content
blank page, kitten images
mediaserver
Parse crash.mp4
Crash
crashed by crash.mp4
Wait for mediaserver...
Restart
restarted automatically by init


Request leak.mp4
Device ID
Iteration number
(guessed ASLR slide)
Generate leak.mp4
embed:
leak address
guessed offset
from libc.so
Devices Table
module identifier to base address
Parse leak.mp4
Parse fake metadata
Return data over duration field
No
ELF header?
Yes


No
ELF header?
Yes
Request shellcode.mp4
Code executing media
Device ID
Module address
Module identifier
Generate rce.mp4
embed:
libc.so base
gadgets offsets
predicted address
shellcode
libc.so base
Devices Table
gadget offsets
predicted addresses
module identifier to base address
Parse rce.mp4
Code execution!
Privilege escalation

Summary

- Requires prior knowledge about the victim's device
  - Further exploits might be used to get this information

- Look-up tables are key information for the exploit
  - Further research to lay aside all tables → more generic exploit

"Even though a universal exploit with no prior knowledge was not achieved, because it is necessary to build lookup tables per ROM, it has been proven practical to exploit in the wild."