Forging golden hammer against Android software protection tools

A deep dive inside anti-reverse & universal bypass with Frida

https://github.com/frenchyeti/unrasp

Georges-Bastien Michel - March 2022

</ɹǝʌǝʁSense>

Who am I ?

@FrenchYeti

Georges-Bastien Michel

CEO / Security Evaluator / Reverser / Developer / Dad

https://github.com/FrenchYeti

Founder of </ɹǝvǝʞSense>

Author of Dexcalibur, Interruptor, …

Summary

1. Intro
2. Reverse & bypass of security checks
3. Improve hooks loading
4. Conclusion

Why this talk ?

1/ Android Protection tool editors do not offer public security reward program

2/ Assessment of protections is mandatory to obtain some certification

3/ We tried to contact them but they did not answer

Before to continue..

Notice : Most of the reverse engineering tasks have been done during security assessment of protected apps while a strict certification process

Runtime Application Security Protection editors implement a lot of heuristics and improve it continuously, so some tricks will be probably out dated in the future.

1/ Intro

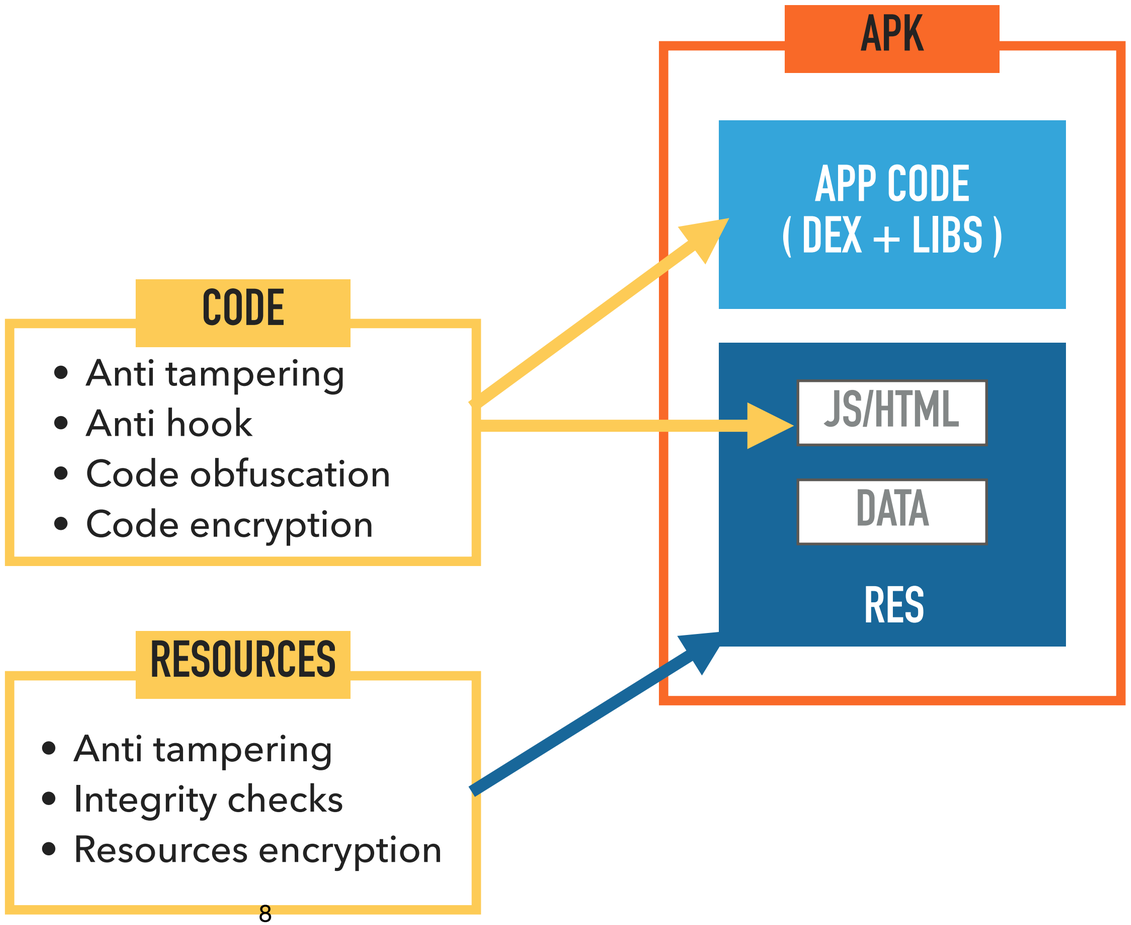
SPT = Software Protection Tools

RASP = Runtime Application Security Protection

SVC = SuperVisor Call = sys call interrupts for arm64

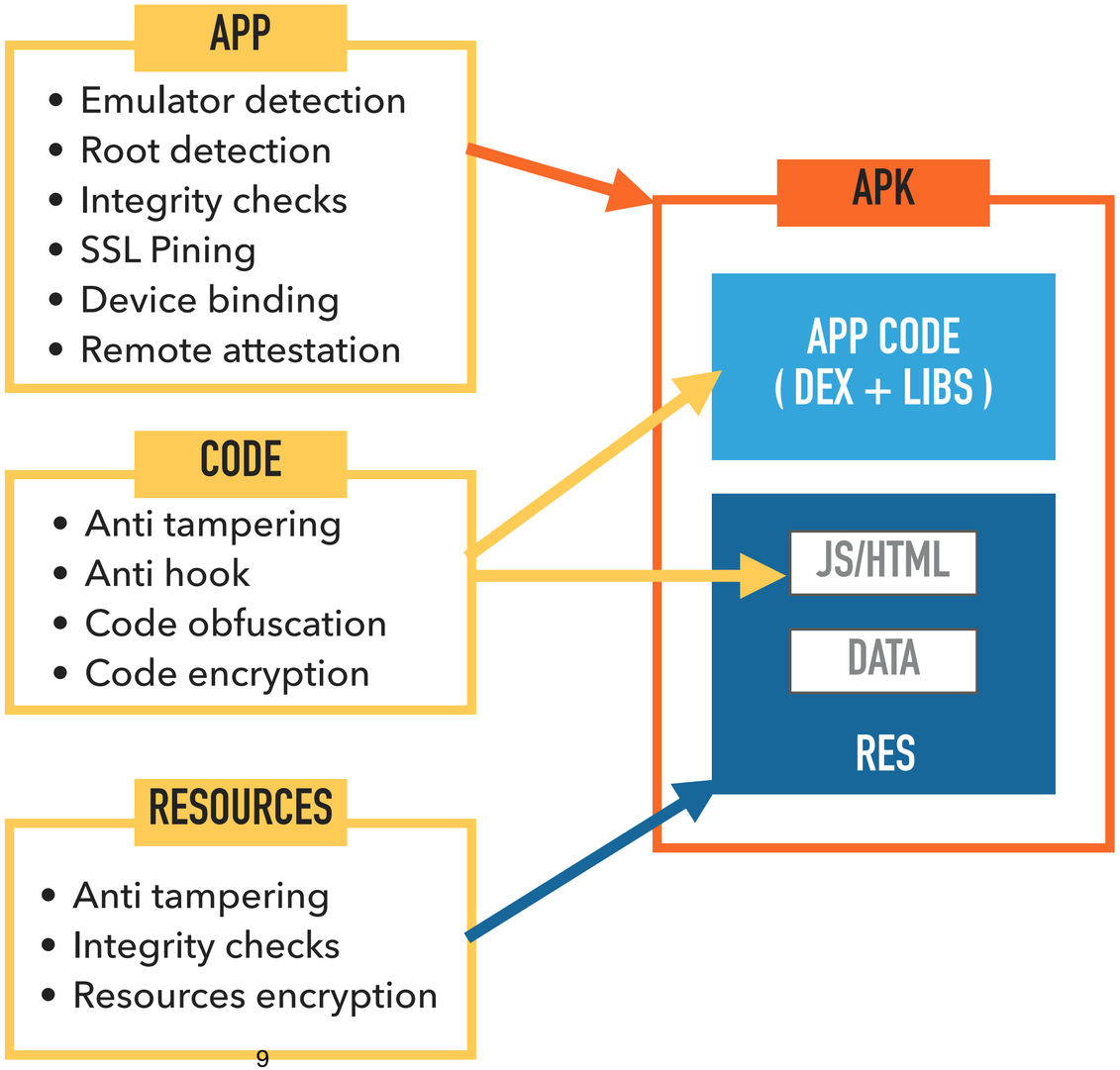
What is a software protection tool ?

Protections layers



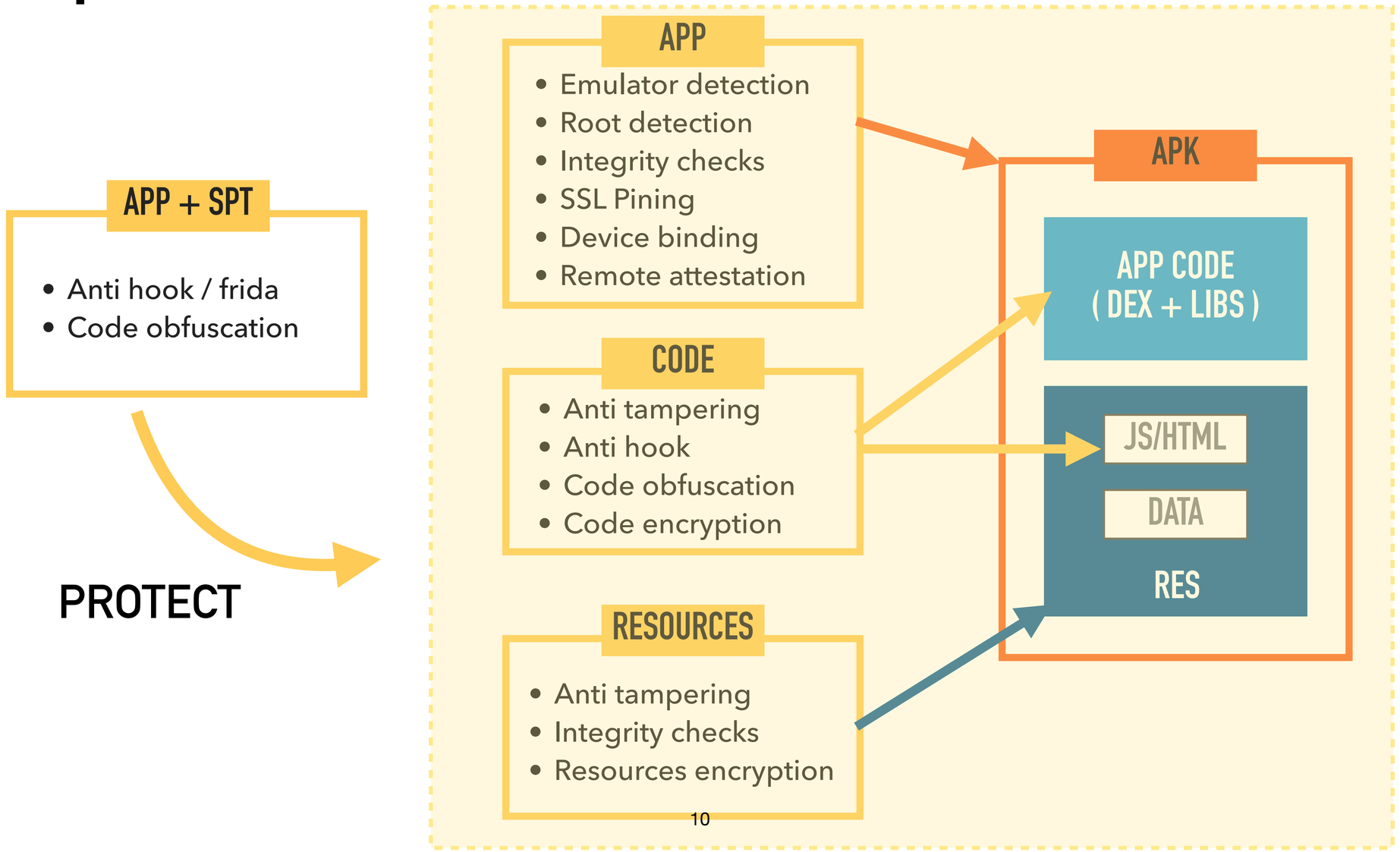
What is a software protection tool ?

Protections layers



What is a software protection tool ?

Top level protections



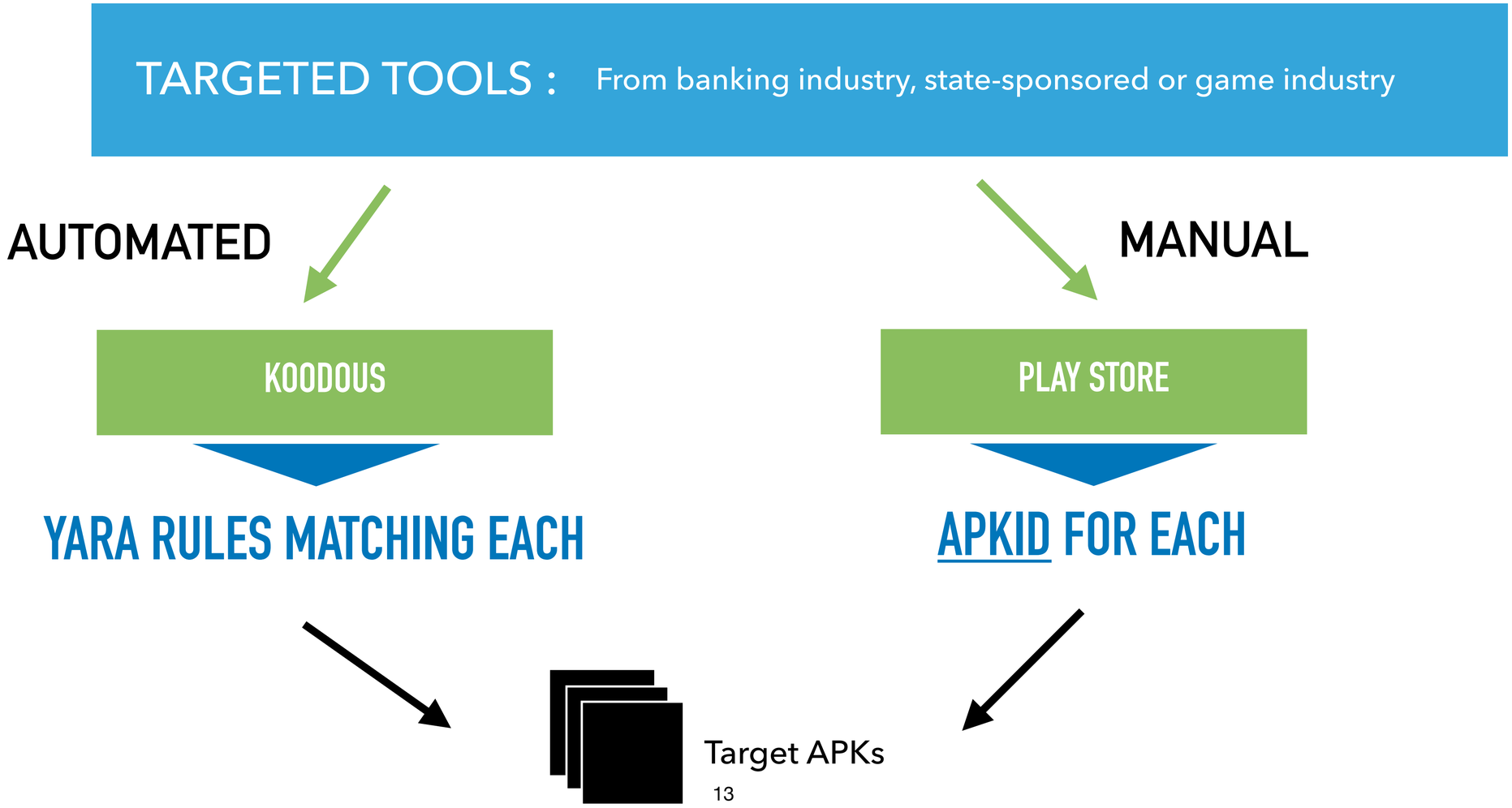
To resume …

Security functions to bypass during tests

- **Code**
  - Java string/classes encryption
  - Anti-debug (gdb)
  - Code encryption
  - Code obfuscation
  - Cross-references hiding
  - Anti-emulator / anti-code lifting
  - Whitebox crypto
  - Code virtualization
- **Content**
  - Assets encryptions
- **Environment**
  - Hook detection
  - Root / Jailbreak detection
  - Custom ROM detection
  - Emulator detection
  - Debug detection
- **Network**
  - SSL Pinning
  - Rogue-certificate detection
  - Network Security monitoring
- **Integrity**
  - Standalone APK integrity checks
  - Certificate checks
  - Anti-tampering of control flow
- **Runtime**
  - Periodic + on-the-fly environment checks

APK / RASP sampling

APK / RASP sampling



APK / RASP sampling

```
  author      = "FrenchYeti"
strings:
  [redacted]
  $hook = {
    0b 1d 00 12  //  and         w11,bf,#0xff
    48 15 40 38  //  ldrb        bf,[x10], #0x1
    29 25 1b 53  //  ubfiz       w9,w9,#0x5,#0xa
    29 01 0b 4a  //  eor         w9,w9,w11
    88 ff ff 35  //  cbnz        bf,LAB_00106e44
    e8 c1 86 52  //  mov         bf,#0x360f
    3f 01 08 6b  //  cmp         w9,bf
  }
  // recurring patterns used into several string decryption
  $str = {
    6c 69 69 38  //  ldrb        w12,[x11, x9, LSL ]
    8c ?? ?? 11  //  add         w12,w12,??
    6c 69 29 38  //  strb        w12,[x11, x9, LSL ]
    29 05 00 91  //  add         x9,x9,#0x1
    3f ?? ?? f1  //  cmp         x9,??
    ec 17 9f 1a  //  cset        w12,??
  }
  $str2 = {
    30 ?? cc 9b 10 fe ?? d3 10 a6 0d 9b 6f 69 69 38 d0 69 70 38
    0f 02 0f 4a 6f 69 29 38 29 05 00 91 3f ?? ?? f1 ef 17 9f 1a
  }
  // binaries have always 8 svc instructions
  $svc = {
    ?8 ?? ?? d2  //  mov         x8,??
    01 00 00 d4  //  svc         0x0
    1f 04 40 b1  //  cmn         x0, #0x1, LSL#12
    00 94 80 da  //  cneg        x0, x0, hi
    ?8 ?? ?? 54  //  b.hi        ??
    c0 03 5f d6  //  ret
  }

condition:
  $hook and ($str or $str2) and #svc >= 6
```

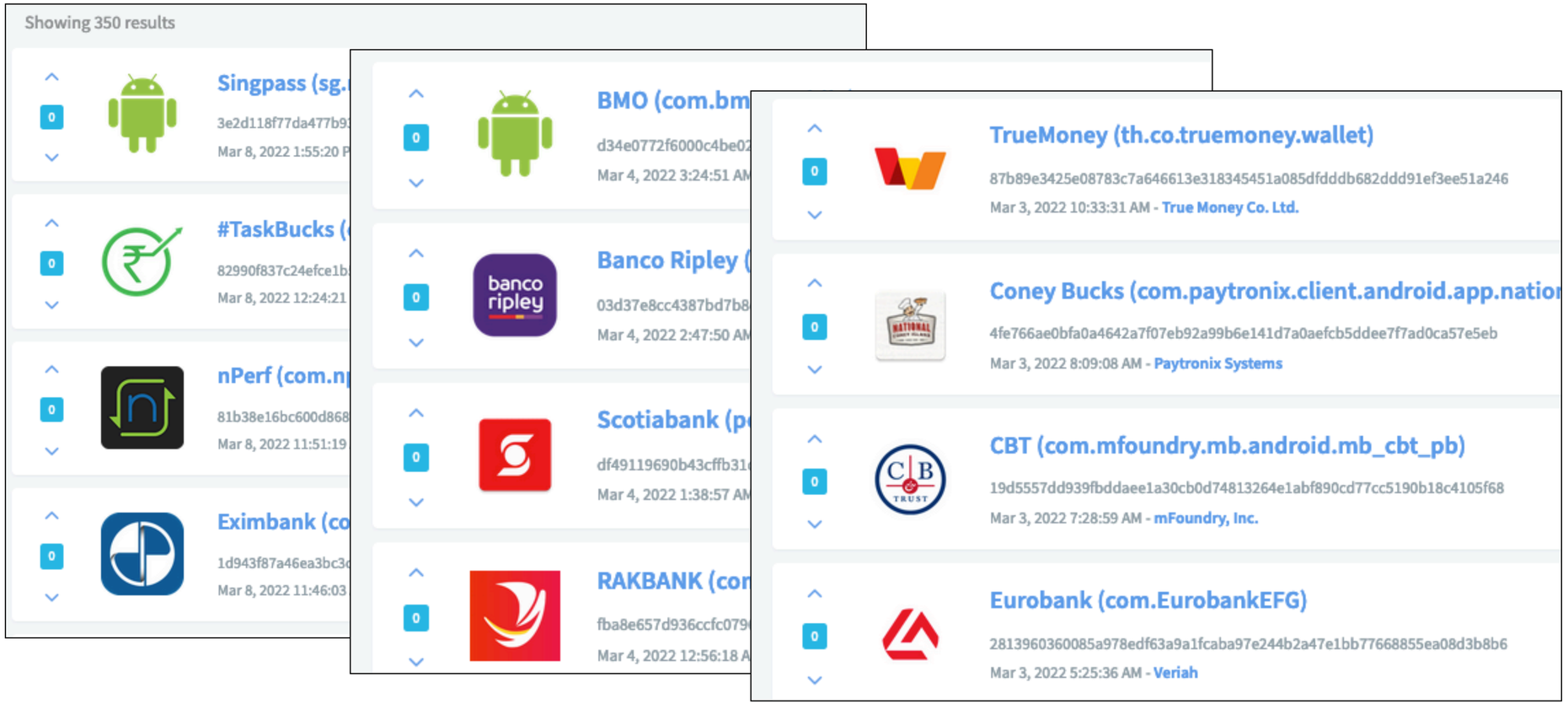
🔥 **No string-based patterns**

Called by parser of /proc/%d/maps content

2 string decryption patterns

Wrapped system call (SVC)

APK / RASP sampling



~350 APKs
for
1 SPT

RASP Overview

Runtime Protection

RASP Overview

Workflow

RASP Entrypoint

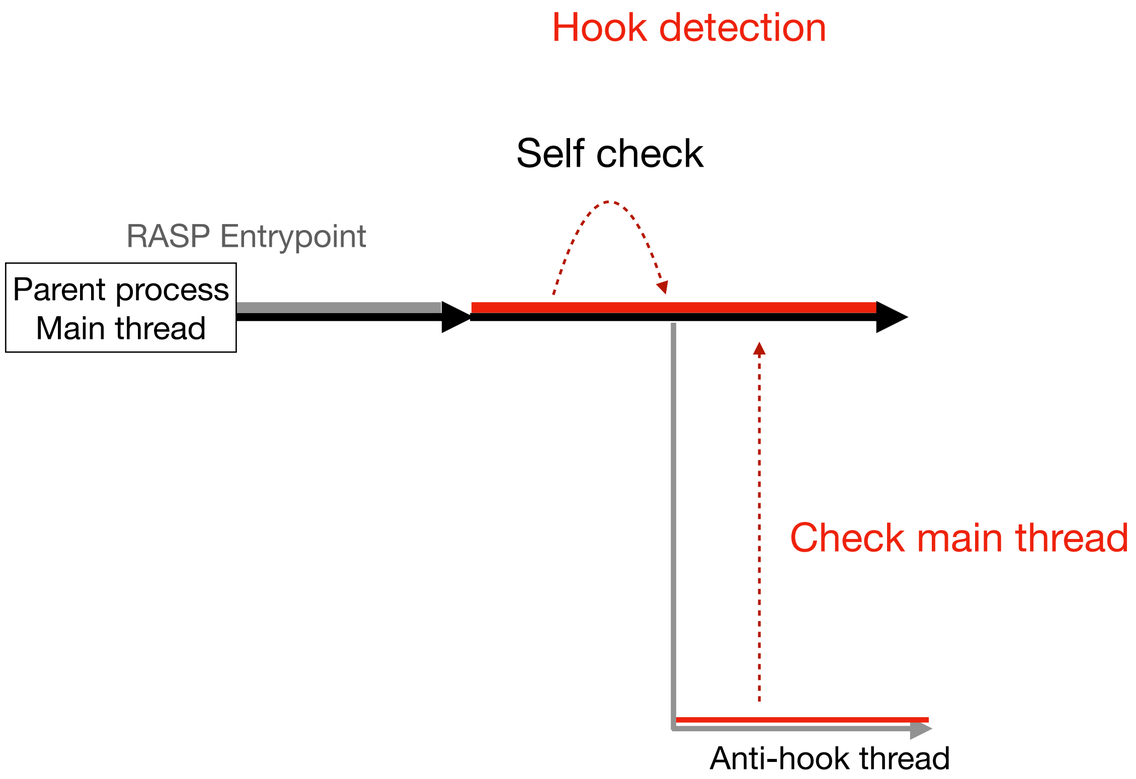
- .init_array[0]
- JNI_OnLoad

thread

fork

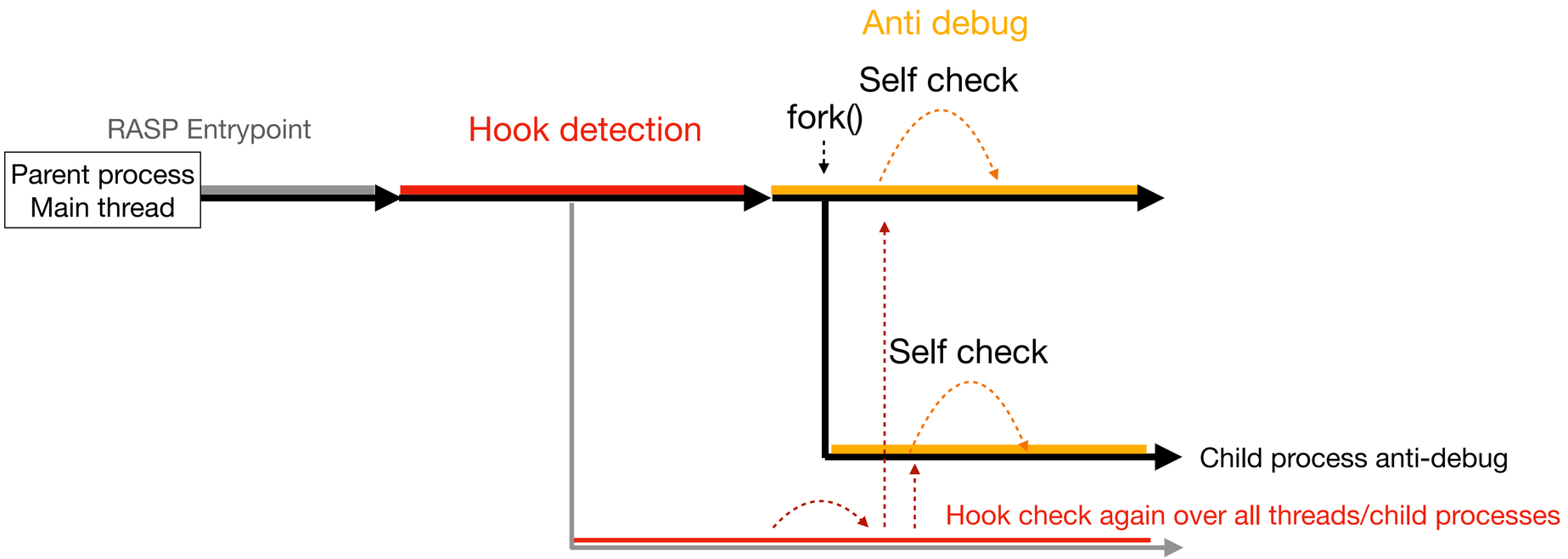
RASP Overview

Workflow



RASP Overview

Workflow

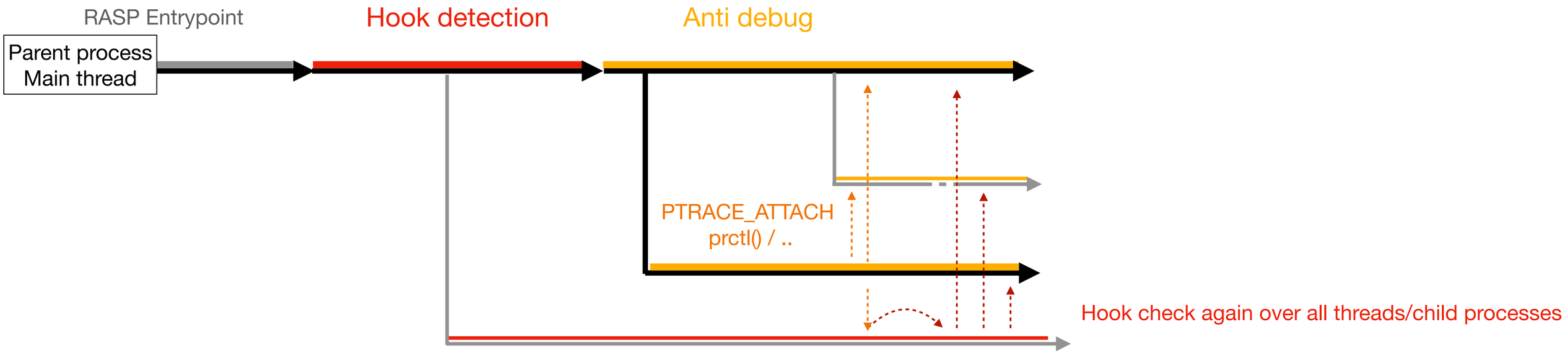


thread

fork

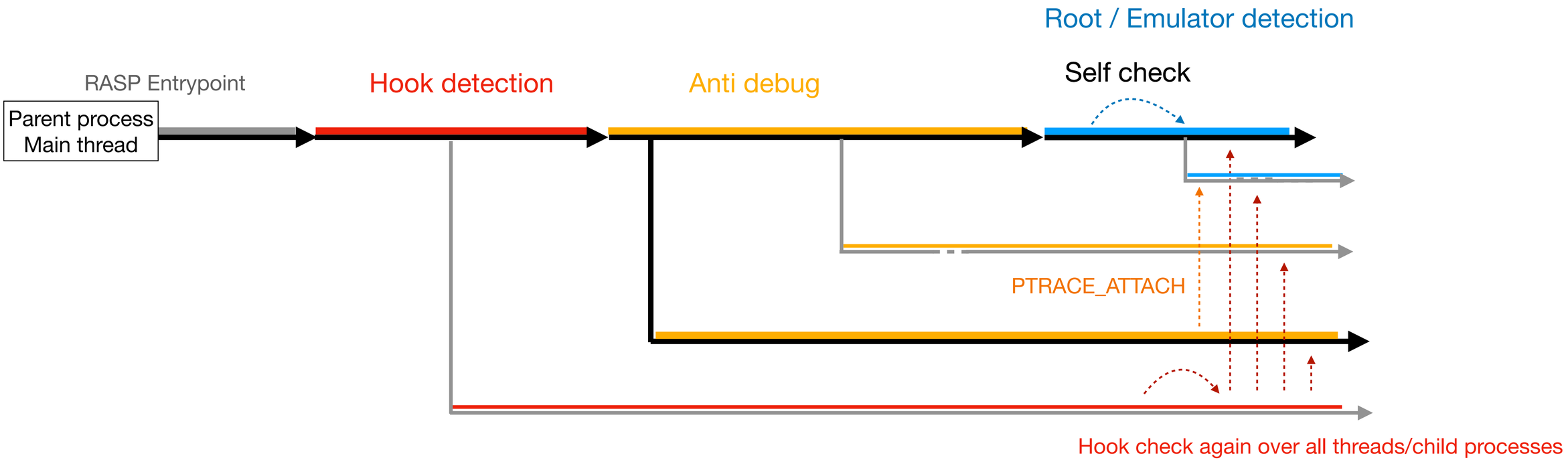
RASP Overview

Workflow



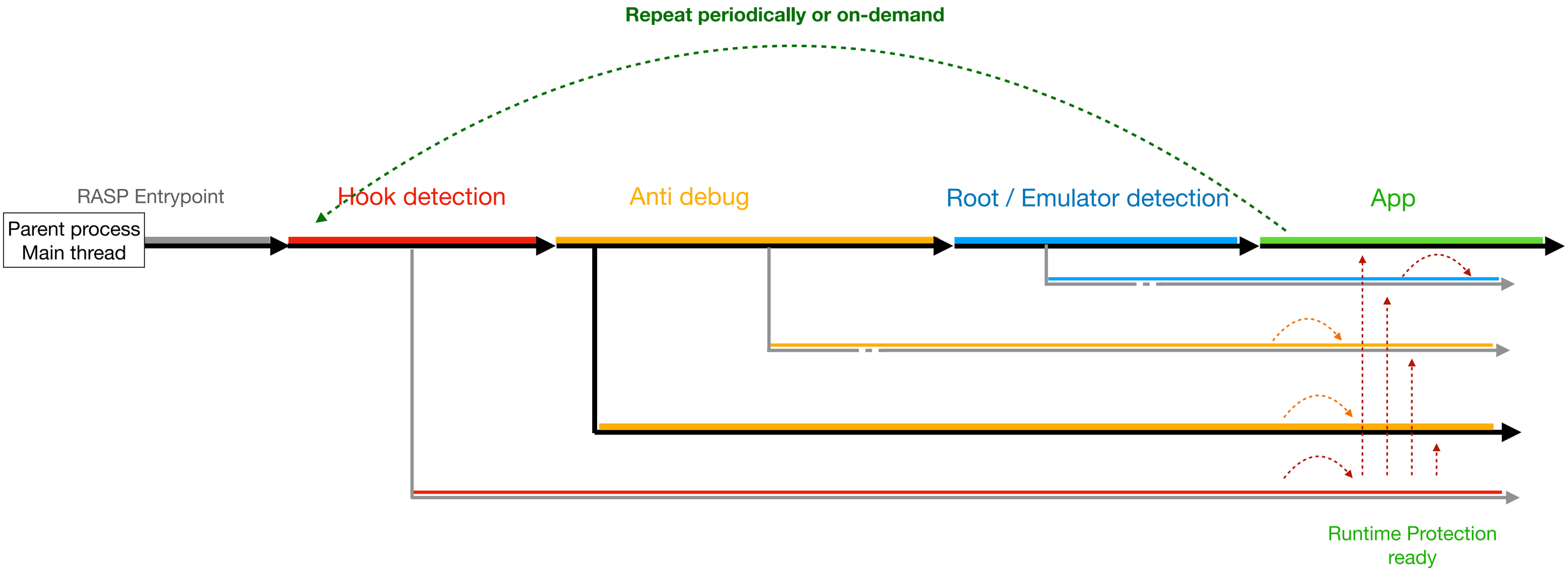
RASP Overview

Workflow



RASP Overview

Workflow



2/
Reverse of some certified SPTs

Tool : Interruptor

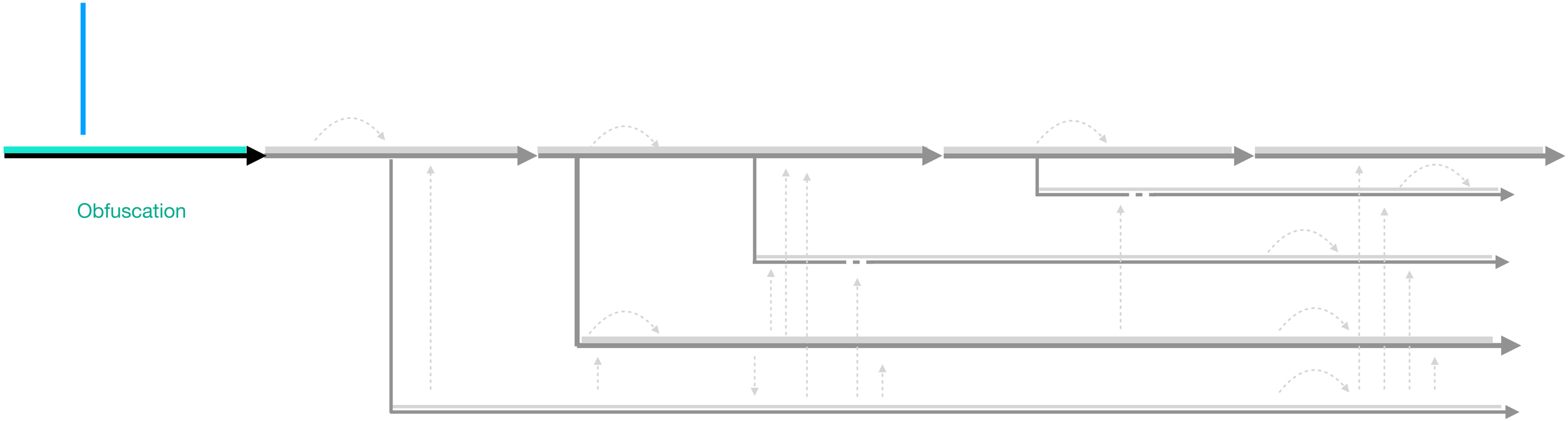
Strace + enhanced SVC hooking for Frida

```
Interruptor.newAgentTracer({
    exclude: {
        syscalls: ["clock_gettime"]
    },
    svc: {
        read: {
            onLeave: function(ctx){
                let res = Memory.scanSync(ctx.x1, ctx.x2.toInt32(), Interruptor.utils().toScanPattern('frida'));
                if(res.length > 0){
                    res.map( m => m.address.writeByteArray([0x41,0x41,0x41,0x41,0x41]));
                    console.log("remove 'frida' pattern from resulting buffer");
                }
            }
        }
    }
}).startOnLoad(/<YOUR_LIB>/g);
```

(strace + sys call hook + coverage + rich API) lib for Frida

https://github.com/FrenchYeti/interruptor

Obfuscation
&
RASP Entrypoint



thread

fork

Hash-based pattern

Hash-based patterns

String-based pattern

Identifiable
List of strings to spoof/patch

Hooks ⚡

```
static const char *APPNAME = "DetectFrida";
static const char *FRIDA_THREAD_GUM_JS_LOOP = "gum-js-loop";
static const char *FRIDA_THREAD_GMAIN = "gmain";
static const char *FRIDA_NAMEDPIPE_LINJECTOR = "linjector";
static const char *PROC_MAPS = "/proc/self/maps";
static const char *PROC_STATUS = "/proc/self/task/%s/status";
static const char *PROC_FD = "/proc/self/fd";
static const char *PROC_TASK = "/proc/self/task";
```

@DarvinciSec's DetectFrida

```
static final String[] knownRootAppsPackages = {
        "com.noshufou.android.su",
        "com.noshufou.android.su.elite",
        "eu.chainfire.supersu",
        "com.koushikdutta.superuser",
        "com.thirdparty.superuser",
        "com.yellowes.su",
        "com.topjohnwu.magisk",
        "com.kingroot.kinguser",
        "com.kingo.root",
        "com.smedialink.oneclickroot",
        "com.zhiqupk.root.global",
        "com.alephzain.framaroot"
};

public static final String[] knownDangerousAppsPackages = {
        "com.koushikdutta.rommanager",
        "com.koushikdutta.rommanager.license",
        "com.dimonvideo.luckypatcher",
        "com.chelpus.lackypatch",
        "com.ramdroid.appquarantine",
        "com.ramdroid.appquarantinepro",
        "com.android.vending.billing.InAppBillingService.COIN",
        "com.android.vending.billing.InAppBillingService.LUCK",
        "com.chelpus.luckypatcher",
```

RootBeer patterns

Hash-based patterns

Retrieving strings thanks to p-code emulation

```
thread = cRead_8(tpidr_el0);
threadStack = *(long *)(thread + 0x28);
l = 0;
do {
  local_a4[l] = (&mask_v_android.permission.REQUEST_INSTALL_PACKAGES)[l] ^
                (&mask_RootDetection_package)[l];
  l = l + 1;
} while (l != 0x2b);
l = 0;
local_79 = 0;
do {
  local_c0[l] = (&mask_v_android.permission.CAMERA)[l] ^ (&mask_RootDetectio
  l = l + 1;
} while (l != 0x19);
l = 0;
local_a7 = 0;
```

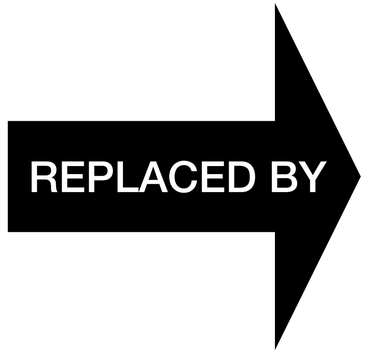
```
Emulation from done ![redacted].py> Finished!
[redacted].py> Running...
x11 = 6291456
Decoded :  magisk\x00\x00\x00\x00`\x00\x00\x00\x00\x00\x00\x00\x00\x00\
Emulation complete ( 0x0024d32c ). At 0xNone (len:96) : None

Emulation from done ![redacted].py> Finished!
[redacted].py> Running...
x11 = 6291456
Decoded :  @eu.chainfire.supersu\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
Emulation complete ( 0x0024bc1c ). At 0xNone (len:96) : None

Emulation from done ![redacted].py> Finished!
[redacted].py> Running...
x11 = 6291456
Decoded :  com.amphoras.hidemyroot.com.android.engineeringmode.com.deva
Emulation complete ( 0x0024ac28 ). At 0xNone (len:96) : None

Emulation from done ![redacted].py> Finished!
[redacted].py> Running...
x11 = 6291456
Decoded :  android.permission.REQUEST_INSTALL_PACKAGES
Decoded :  android.permission.CAMERA
Decoded :  android.permission.READ_CONTACTS
Decoded :  android.permission.REQUEST_INSTALL_PACKAGES\x00\x00\x00\x00\
Emulation complete ( 0x0024b2c8 ). At 0xNone (len:96) : None

Emulation from done ![redacted].py> Finished!
```

Hash-based patterns

Old

Modern

Search
Compare
with strings



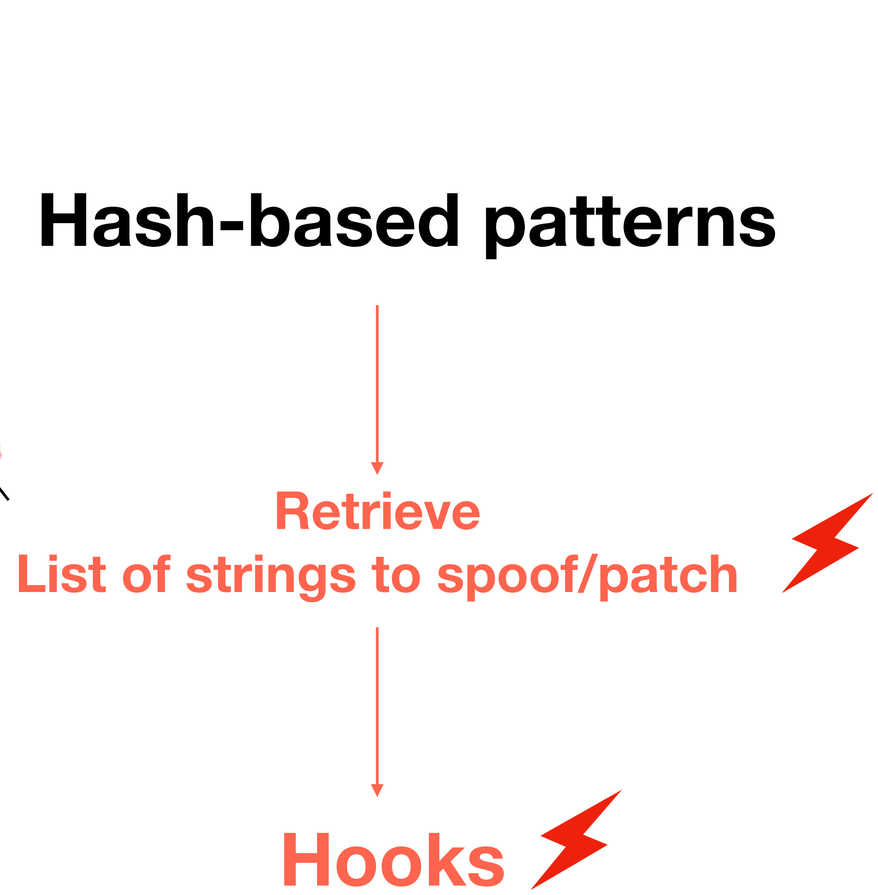
Enumerate
Hash
Compare

Hash-based patterns

- **Improvements :**

  - Prevent to retrieve all patterns in clear text at runtime
  - Easier to flatten and inline
  - Less easy to hook (instruction only if well done)
  - Low fingerprint on Android API

Hash-based patterns



Hash-based patterns

Open « /vendor/etc/public.libraries.txt »

String-based pattern

Lib C


```
fopen( /vendor/etc/public.libraries.txt, ..)
```

Hash-based pattern

SVC


```
(SVC)  openat ( AT_FDCWD, /system/etc/ , …) > 47
(SVC)  getdents64 (  fd = 47  /system/etc/ ,…)

Loop of each dir entries of type ‘file’ :

  hash_16_first_bytes(  entry->d_name )
     hash == HASH_public.libraries
  hash_3_last_bytes(  entry->d_name )
     hash == HASH_txt
          True
(SVC)  openat (  AT_FDCWD,  entry->d_name ,…)
```

False

True

Hash-based patterns

Open « /vendor/etc/public.libraries.txt »

```
  free(pcVar5);
}
else {
  uVar2 = char_xor_0_or_offset(0,*pcVar5);
  if (pcVar5[1] != '\0') {
    uVar2 = char_xor_0_or_offset(uVar2,pcVar5[1]);
    if (pcVar5[2] != '\0') {
      uVar2 = char_xor_0_or_offset(uVar2,pcVar5[2]);
      if (pcVar5[3] != '\0') {
        uVar2 = char_xor_0_or_offset(uVar2,pcVar5[3]);
        if (pcVar5[4] != '\0') {
          uVar2 = char_xor_0_or_offset(uVar2,pcVar5[4]);
          if (pcVar5[5] != '\0') {
            uVar2 = char_xor_0_or_offset(uVar2,pcVar5[5]);
            if (pcVar5[6] != '\0') {
              uVar2 = char_xor_0_or_offset(uVar2,pcVar5[6]);
              if (pcVar5[7] != '\0') {
                uVar2 = char_xor_0_or_offset(uVar2,pcVar5[7]);
                if (pcVar5[8] != '\0') {
                  uVar2 = char_xor_0_or_offset(uVar2,pcVar5[8]);
                  if (pcVar5[9] != '\0') {
                    uVar2 = char_xor_0_or_offset(uVar2,pcVar5[9]);
                    if (pcVar5[10] != '\0') {
                      uVar2 = char_xor_0_or_offset(uVar2,pcVar5[10]);
                      if (pcVar5[0xb] != '\0') {
                        uVar2 = char_xor_0_or_offset(uVar2,pcVar5[0xb]);
                        if (pcVar5[0xc] != '\0') {
                          uVar2 = char_xor_0_or_offset(uVar2,pcVar5[0xc]);
                          if (pcVar5[0xd] != '\0') {
                            uVar2 = char_xor_0_or_offset(uVar2,pcVar5[0xd]);
                            if (pcVar5[0xe] != '\0') {
                              uVar2 = char_xor_0_or_offset(uVar2,pcVar5[0xe]);
                              if (pcVar5[0xf] != '\0') {
                                uVar2 = char_xor_0_or_offset(uVar2,pcVar5[0xf]);
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
```

Hash-based pattern

SVC

(SVC) openat (AT_FDCWD, /system/etc/ , …) > 47
(SVC) getdents64 (fd = 47 /system/etc/ ,…)

Loop of each dir entries of type ‘file’ :

hash_16_first_bytes(entry->d_name)

hash == HASH_public.libraries

hash_3_last_bytes(entry->d_name)

hash == HASH_txt

True

False

True

(SVC) openat (AT_FDCWD, entry->d_name ,…)

Hash-based patterns

```
Interruptor.newAgentTracer({
    followThread: false,
    exclude: { syscalls: [/gettime/,/linkat/,/madvise/,/mprotect/], },
    output: {...},
    svc:{...},
    onStart: function(modName, ctor){

        const lib = Process.findModuleByName( name: '<[redacted]_LIB>');

        let LAST_X_HASH = null;
        let LAST_X_STR = null;
        Interceptor.attach(lib.base.add(0x313C), callbacksOrProbe: {
            onEnter: function(args : InvocationArguments ){
                this.c = args[0].toUInt32();
                if(this.c==0){
                    if(LAST_X_HASH != null){
                        console.error(` [HOOK] [redacted] hash_x_20 (${LAST_X_STR}) => 0x ${LAST_X_HASH.toString(16)}`);
                    }
                    LAST_X_STR = String.fromCharCode(args[1].toUInt32())
                }else{
                    LAST_X_STR += String.fromCharCode(args[1].toUInt32())
                }
            },
            onLeave: function(ret : InvocationReturnValue ){
                LAST_X_HASH = ret.toUInt32();
            }
        });
```

Trace all syscalls, hook a hash function and store hashed byte (1-by-1) until a new hash computation starts (not supports concurrent exec)

Hash-based patterns

```
openat (  dfd = AT_FDCWD  , filename = /system/etc/ , int flags = 0x84000 , mode
getdents64 (  fd = 41  /system/etc/   , struct linux_dirent64 *dirent = 0x753f22
    hash_x_20 (apns-conf.xml) => 0x33c0a3c9
    hash_x_20 (fs_config_dirs) => 0x95c227d2
    hash_x_20 (default-permissi) => 0x89d8d51a
    hash_x_20 (treble_sepolicy_) => 0xcf692f07
    hash_x_20 (hosts) => 0x6b642f3
    . . .
    hash_x_20 (init) => 0x353554
    hash_x_20 (mke2fs.conf) => 0x6569352f
    hash_x_20 (media_profiles_V) => 0xf5024759
    hash_x_20 (ipsec.secrets) => 0xa5c85899
    hash_x_20 (treble_sepolicy_) => 0xcf692f07
openat ( dfd = AT_FDCWD, filename = /system/etc/public.libraries-trustonic.txt ,
read ( fd = 42  /system/etc/public.libraries-trustonic.txt , buf = 0x7fcf587084
    hash_x_20 (public.libraries) => 0x97229a01
read ( fd = 42  /system/etc/public.libraries-trustonic.txt , buf = 0x7fcf587084
close ( fd = 42  /system/etc/public.libraries-trustonic.txt )> 0x0
```

Hash-based pattern

SVC


```
(SVC)  openat ( AT_FDCWD, /system/etc/ , …) > 47
(SVC)  getdents64 (  fd = 47  /system/etc/ ,…)
                    Loop …
(SVC)  openat (  AT_FDCWD,  entry->d_name ,…)
```

Require Stalker / Interruptor

SVC hook with
multi-threads + clone
support

Breaking Stack analysis

Anti-decompiler

Breaking stack analysis

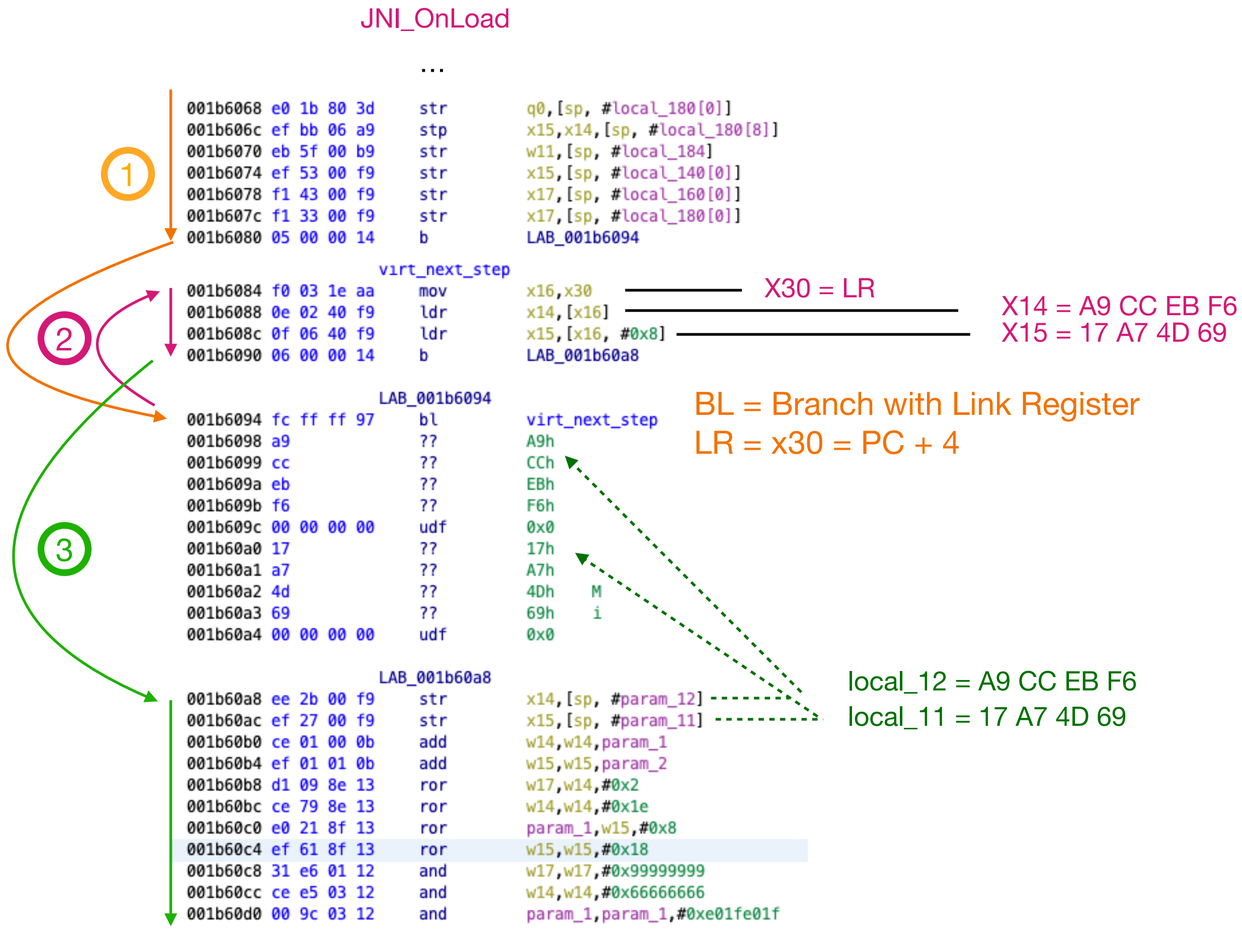
```
001b6068 e0 1b 80 3d     str        q0,[sp, #local_180[0]]
001b606c ef bb 06 a9     stp        x15,x14,[sp, #local_180[8]]
001b6070 eb 5f 00 b9     str        w11,[sp, #local_184]
001b6074 ef 53 00 f9     str        x15,[sp, #local_140[0]]
001b6078 f1 43 00 f9     str        x17,[sp, #local_160[0]]
001b607c f1 33 00 f9     str        x17,[sp, #local_180[0]]
001b6080 05 00 00 14     b          LAB_001b6094

                     virt_next_step
001b6084 f0 03 1e aa     mov        x16,x30
001b6088 0e 02 40 f9     ldr        x14,[x16]
001b608c 0f 06 40 f9     ldr        x15,[x16, #0x8]
001b6090 06 00 00 14     b          LAB_001b60a8

                     LAB_001b6094
001b6094 fc ff ff 97     bl         virt_next_step
001b6098 a9              ??         A9h
001b6099 cc              ??         CCh
001b609a eb              ??         EBh
001b609b f6              ??         F6h
001b609c 00 00 00 00     udf        0x0
001b60a0 17              ??         17h
001b60a1 a7              ??         A7h
001b60a2 4d              ??         4Dh    M
001b60a3 69              ??         69h    i
001b60a4 00 00 00 00     udf        0x0

                     LAB_001b60a8
001b60a8 ee 2b 00 f9     str        x14,[sp, #param_12]
001b60ac ef 27 00 f9     str        x15,[sp, #param_11]
001b60b0 ce 01 00 0b     add        w14,w14,param_1
001b60b4 ef 01 01 0b     add        w15,w15,param_2
001b60b8 d1 09 8e 13     ror        w17,w14,#0x2
001b60bc ce 79 8e 13     ror        w14,w14,#0x1e
001b60c0 e0 21 8f 13     ror        param_1,w15,#0x8
001b60c4 ef 61 8f 13     ror        w15,w15,#0x18
001b60c8 31 e6 01 12     and        w17,w17,#0x99999999
001b60cc ce e5 03 12     and        w14,w14,#0x66666666
001b60d0 00 9c 03 12     and        param_1,param_1,#0xe01fe01f
```

```
 4 void startupCheck(int int1,int int2)
 5
 6 {
 7   uint uVar1;
 8   uint uVar2;
 9   long unaff_x29;
10   ulong *unaff_x30;
11   ulong uStack0000000000000048;
12   ulong uStack0000000000000050;
13
14   uVar1 = (int)*unaff_x30 + int1;
15   uVar2 = (int)unaff_x30[1] + int2;
16   uStack0000000000000050 =
17        *unaff_x30 & 0xffffffff00000000 |
18        (ulong)((uVar1 >> 2 | uVar1 * 0x40000000) & 0x99999999 |
19                (uVar1 >> 0x1e | uVar1 * 4) & 0x66666666);
20   uStack0000000000000048 =
21        unaff_x30[1] & 0xffffffff00000000 |
22        (ulong)((uVar2 >> 8 | uVar2 * 0x1000000) & 0xe01fe01f |
23                (uVar2 >> 0x18 | uVar2 * 0x100) & 0x1fe01fe0);
24   uStack0000000000000050 = uStack0000000000000050 + ((ulong)unaff_x30 & 0xfffffffffffffffe);
25   uStack0000000000000048 = uStack0000000000000048 + (uStack0000000000000050 & 0xfffffffffffffffe);
26   FUN_001b6148(unaff_x29 + -0xe0,&stack0x000000c0,&stack0x00000044,unaff_x29 + -0x70,
27                unaff_x29 + -0xa8);
28                    /* WARNING: Bad instruction - Truncating control flow here */
29   halt_baddata();
30 }
```

Data inserted inside code

Anti-decompiler

Breaking stack analysis



Basic Dex Class Encryption

Dex Class Encryption

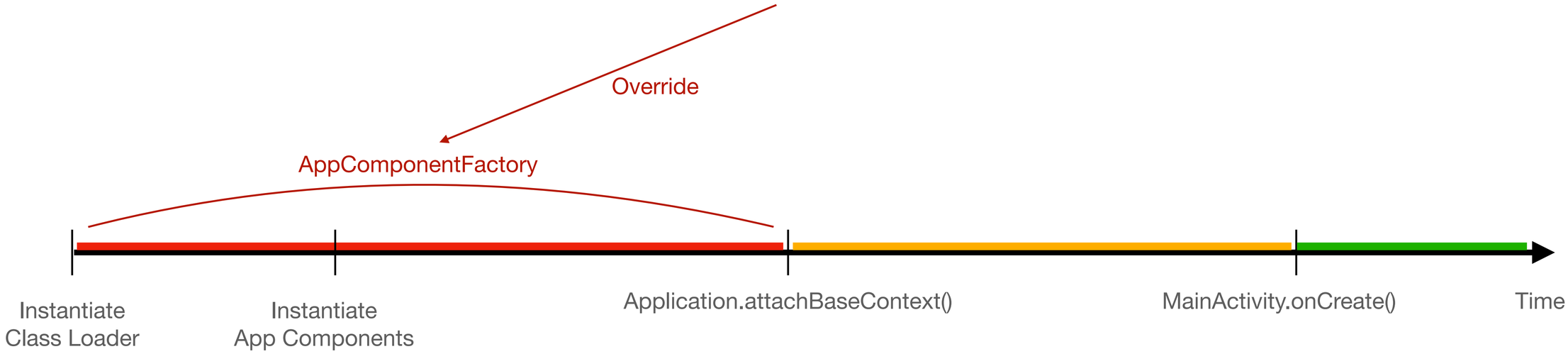
Custom AppComponentFactory

```
<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher"
    android:name="a.m" android:allowBackup="false" android:supportsRtl="true" android:extractNativeLibs="false"
    android:roundIcon="@mipmap/ic_launcher_round" android:appComponentFactory="a.f">
```

Dex Class Encryption

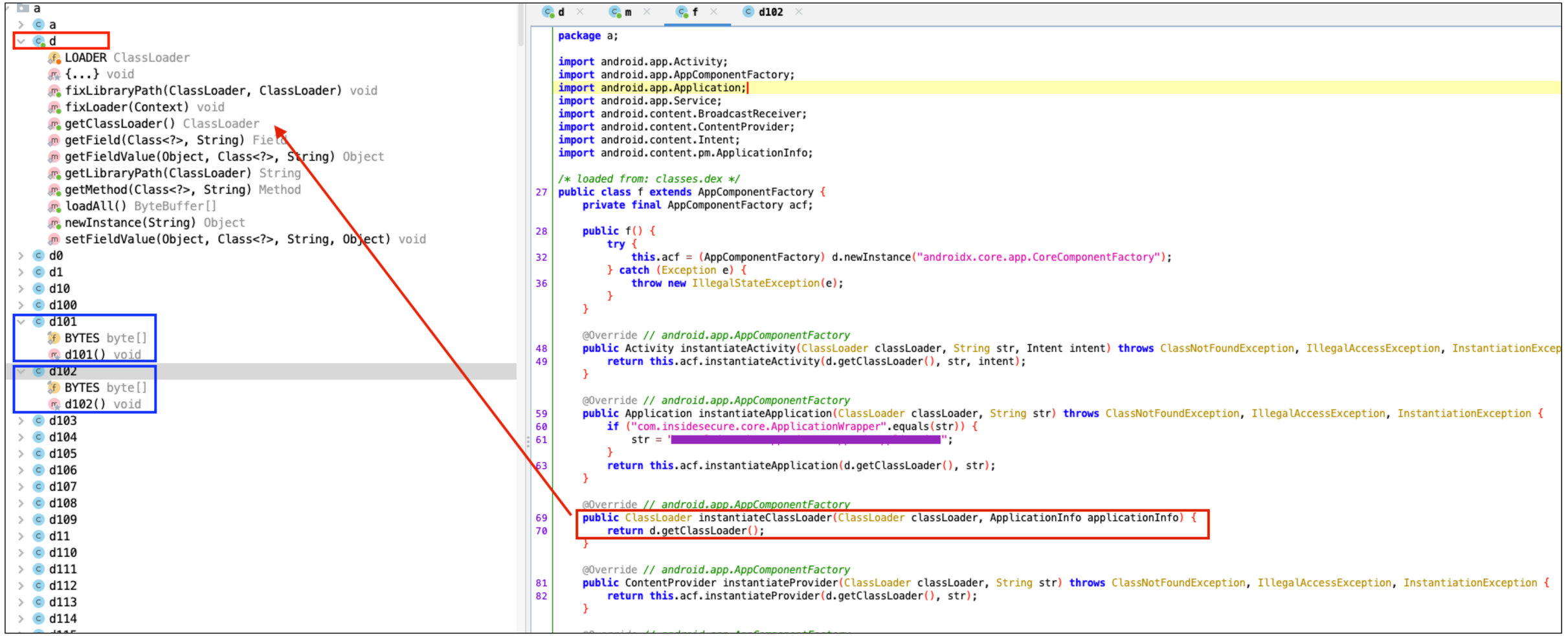
Custom AppComponentFactory

```
<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher"
  android:name="a.m" android:allowBackup="false" android:supportsRtl="true" android:extractNativeLibs="false"
  android:roundIcon="@mipmap/ic_launcher_round" android:appComponentFactory="a.f">
```



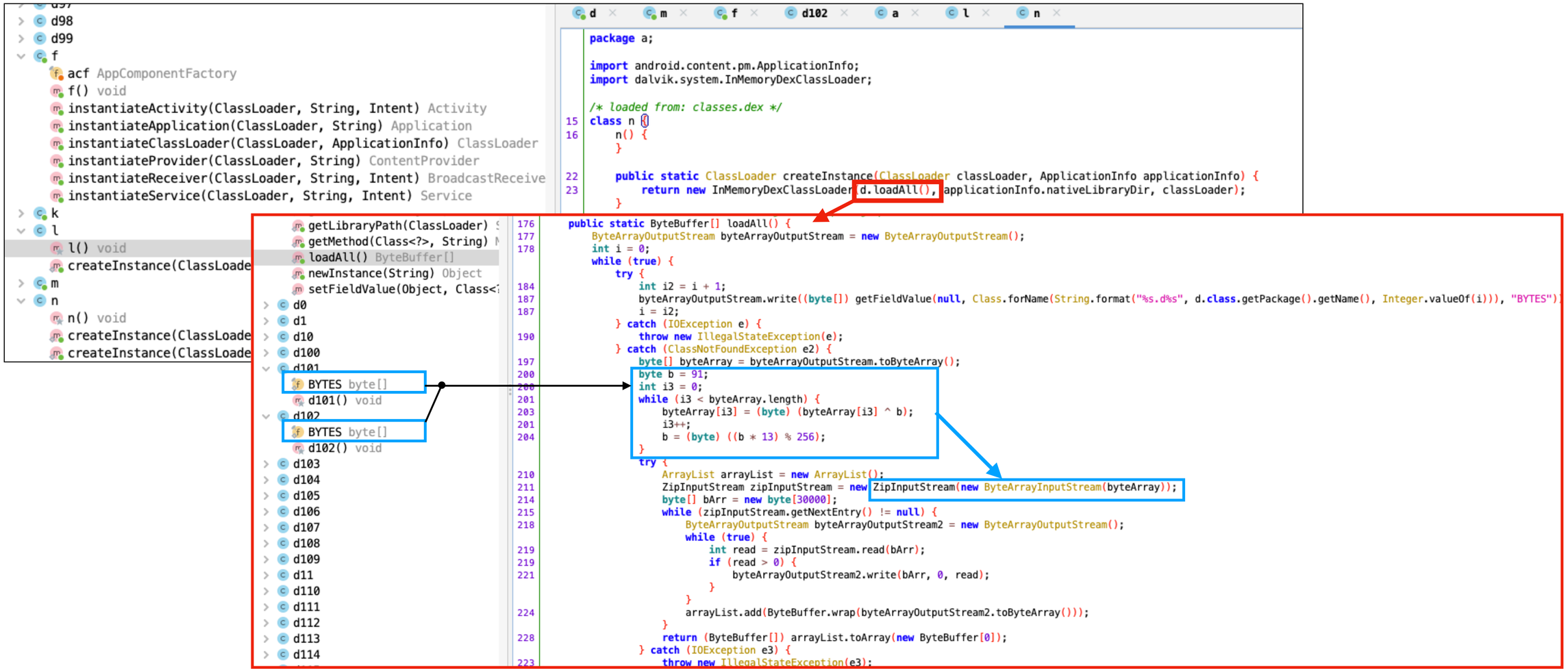
Dex Class Encryption

Custom AppComponentFactory



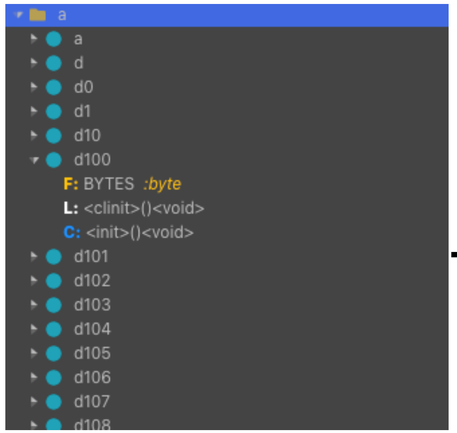
Dex Class Encryption

Custom AppComponentFactory



Dex Class Encryption

Custom AppComponentFactory



930 classes containing each
a xored array of 2048 bytes

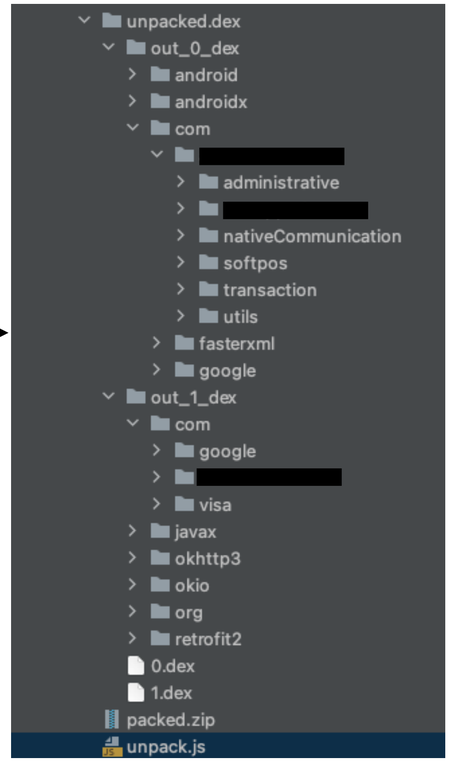
Concatenate each arrays

↓

Xor all bytes

↓

Unzip



Resulting zip contains
2 DEX files

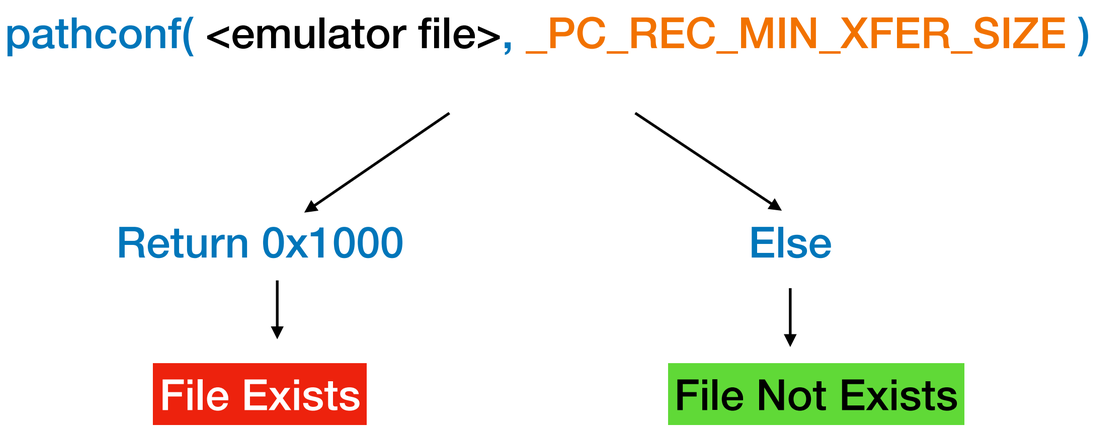
Protected byte code can be retrieved statically

Diversified file checking

File-based detection using pathconf()

```
if ((DAT_001224d0 & 1) == 0) {
  lVar16 = 0;
  cVar12 = '\0';
  while( true ) {
    while (cVar12 == '\x01') {
      DAT_001224d0 = 1;
      cVar12 = '\x02';
    }
    if (cVar12 != '\0') break;
    (&DAT_001222fe_/system/lib/libnoxd.so)[lVar16] =
         (&DAT_001222fe_/system/lib/libnoxd.so)[lVar16] + -0x26;
    lVar16 = lVar16 + 1;
    cVar12 = lVar16 == 0x17;
  }
  if (cVar12 != '\x02') {
    do {
                    /* WARNING: Do nothing block with infinite loop */
    } while( true );
  }
}
lVar16 = pathconf(&DAT_001222fe_/system/lib/libnoxd.so,0xb);
uVar15 = 0xcd;
if (lVar16 != 0x1000) {
  uVar15 = 0;
}
return uVar15;
```

- Use « statfs » sys call instead of « openat »
- Slow down reverse engineering by using not obvious return value



Packed Self-Modifying code

Native packer

Matrioschka

```
0023eae4 fd 03 02 91     add        x29,sp,#0x80
0023eae8 59 d0 3b d5     mrs        x25,tpidr_el0
0023eaec 28 17 40 f9     ldr        x8,[x25, #0x28]
0023eaf0 e8 17 00 f9     str        x8,[sp, #local_68]
0023eaf4 a8 02 00 90     adrp       x8,0x292000
0023eaf8 09 a5 43 b9     ldr        w9,[x8, #0x3a4]=>DAT_002923a4_isInit
0023eafc a9 18 00 35     cbnz       w9,LAB_0023ee10
0023eb00 29 00 80 52     mov        w9,#0x1
0023eb04 8a 02 00 f0     adrp       x10,0x291000
0023eb08 09 a5 03 b9     str        w9,[x8, #0x3a4]=>DAT_002923a4_isInit
0023eb0c 03 00 00 94     bl         do_init
0023eb10 81              ??         81h
0023eb11 f4              ??         F4h
0023eb12 13              ??         13h
0023eb13 00              ??         00h
0023eb14 c2              ??         C2h
0023eb15 50              ??         50h    P
0023eb16 00              ??         00h
0023eb17 00              ??         00h
                         *************************************************************
                         *                                                           *
                         *  FUNCTION                                                 *
                         *************************************************************
                         undefined do_init()
     undefined           w0:1           <RETURN>
                         do_init                                        XREF[1]:     _I
0023eb18 5f f1 4c b9     ldr        wzr,[x10, #0xcf0]
0023eb1c 05 00 00 94     bl         perform_lib_init
0023eb20 40              ??         40h    @
0023eb21 8e              ??         8Eh
0023eb22 01              ??         01h
0023eb23 00              ??         00h
0023eb24 00              ??         00h
0023eb25 00              ??         00h
0023eb26 00              ??         00h
0023eb27 00              ??         00h
0023eb28 ef              ??         EFh
0023eb29 46              ??         46h    F
0023eb2a 02              ??         02h
0023eb2b 00              ??         00h
0023eb2c 00              ??         00h
0023eb2d 00              ??         00h
0023eb2e 00              ??         00h
0023eb2f 00              ??         00h
```

*x30 = 81 f4 13 00

*LR = *x30 = 40 8E 01 00

```
puVar1 = (undefined *)(*unaff_x30 + (long)unaff_x30);
__size = unaff_x30[1];
base = puVar1 + __size;
__addr = (void *)((ulong)puVar1 & 0xfffffffffffff000);
__len = ((ulong)(base + 0xfff) & 0xfffffffffffff000) - (long)__addr;
err = mprotect(__addr,__len,7);
if (err != 0) {
  err_code = (int *)__errno();
  strerror(*err_code);
}
__ptr = (undefined *)malloc(__size);
uVar13 = (uint)__size;
uVar2 = uVar13 & 0xfffffff0;
puVar7 = puVar1;
puVar9 = __ptr;
if (0 < (int)uVar2) {
  lVar15 = 0;
  do {
    FUN_0023cc3c(puVar1 + lVar15,__ptr + lVar15);
    lVar15 = lVar15 + 0x10;
  } while ((int)lVar15 < (int)uVar2);
  puVar7 = puVar1 + lVar15;
  puVar9 = __ptr + lVar15;
}
if ((int)uVar2 < (int)uVar13) {
  uVar13 = uVar13 - uVar2;
  uVar14 = (ulong)uVar13;
  if (uVar13 == 0) {
    FUN_0023cc3c(&stack0x00000018,&stack0x00000008);
  }
  else {
    puVar6 = &stack0x00000018;
    uVar11 = uVar14;
    if ((uVar13 < 0x20) || ((puVar6 < puVar7 + uVar14 && (puVar7 < puVar6 + uVar14)))) {
LAB_0023ebe4:
      do {
        uVar11 = uVar11 - 1;
        *puVar6 = *puVar7;
        puVar6 = puVar6 + 1;
        puVar7 = puVar7 + 1;
      } while (uVar11 != 0);
    }
```

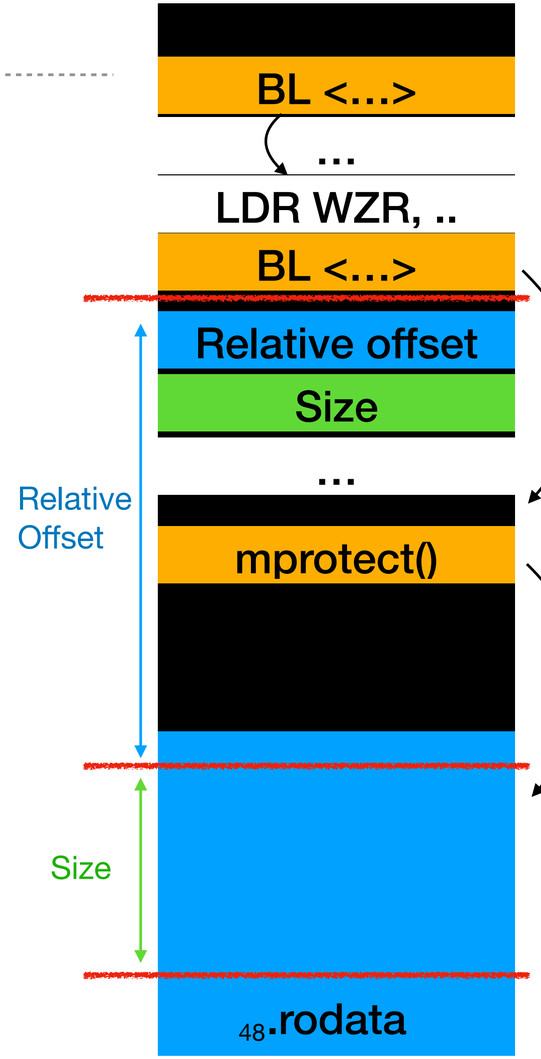
Var = 0x408e01 + 0x13eb20

Point to a location inside .rodata

Native packer

Matrioschka

```
0023eb08 09 a5 03 b9     str        w9,[x8, #0x3a4]=>DAT
0023eb0c 03 00 00 94     bl         do_init
0023eb10 81              ??         81h
0023eb11 f4              ??         F4h
0023eb12 13              ??         13h
0023eb13 00              ??         00h
0023eb14 c2              ??         C2h
0023eb15 50              ??         50h    P
0023eb16 00              ??         00h
0023eb17 00              ??         00h
                         ***********************************
                         *
                         *  FUNCTION
                         ***********************************
                         undefined do_init()
     undefined             w0:1           <RETURN>
                         do_init
0023eb18 5f f1 4c b9     ldr        wzr,[x10, #0xcf0]
0023eb1c 05 00 00 94     bl         perform_lib_init
0023eb20 40              ??         40h    @
0023eb21 8e              ??         8Eh
0023eb22 01              ??         01h
0023eb23 00              ??         00h
0023eb24 00              ??         00h
0023eb25 00              ??         00h
0023eb26 00              ??         00h
0023eb27 00              ??         00h
0023eb28 ef              ??         EFh
0023eb29 46              ??         46h    F
0023eb2a 02              ??         02h
0023eb2b 00              ??         00h
0023eb2c 00              ??         00h
0023eb2d 00              ??         00h
0023eb2e 00              ??         00h
0023eb2f 00              ??         00h
```



1/ x30 = PC+4 -> relative offset

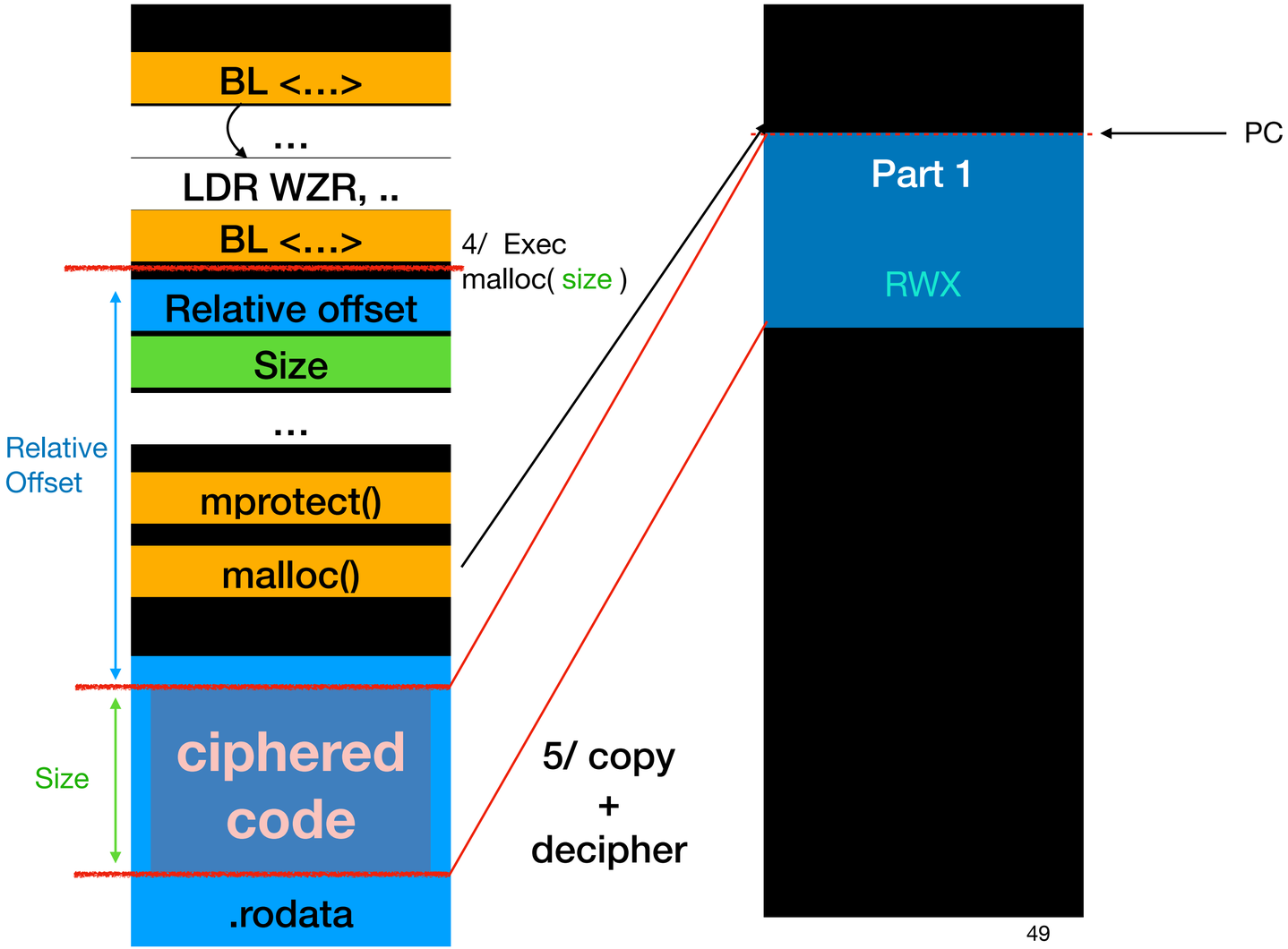
2/ Jump to prologue of mprotect()

3/ mprotect(x30 + relative offset , size , RWX)

BL = Branch with Link Register

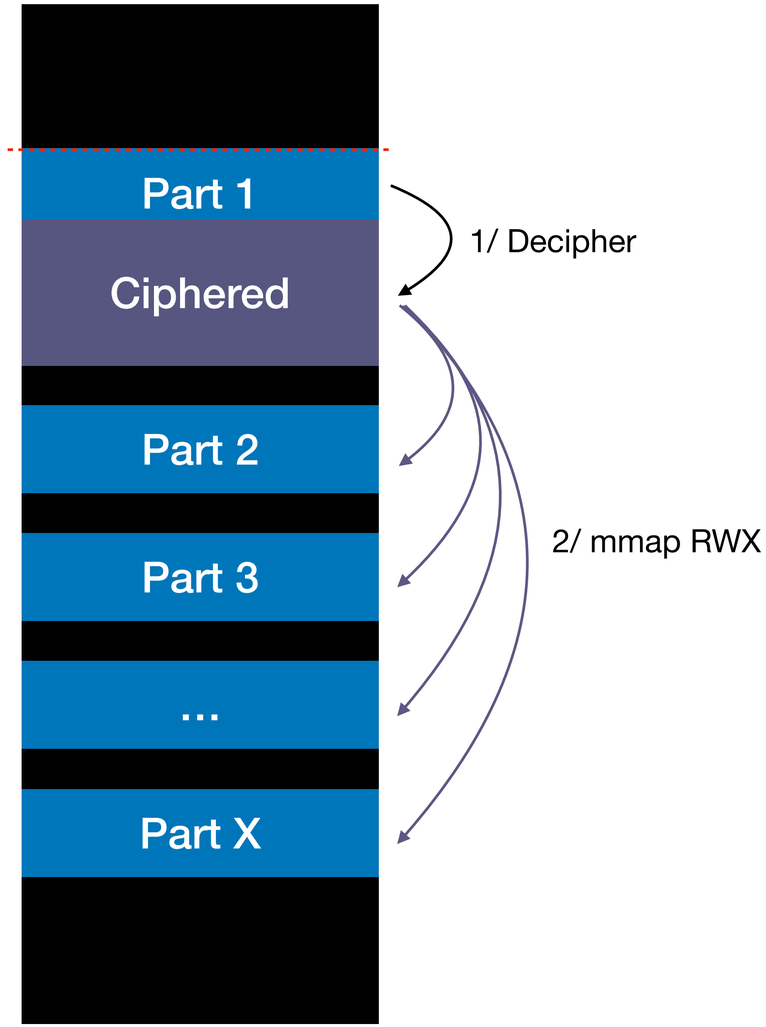
Native packer

Matrioschka



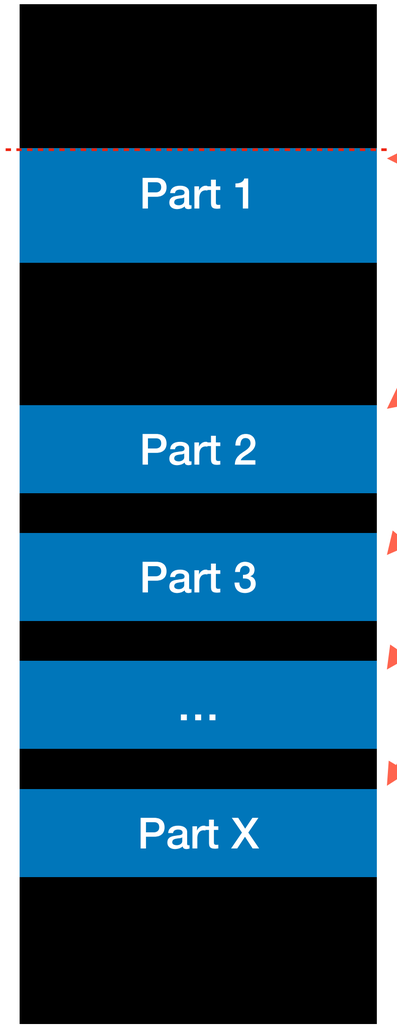
Native packer

Matrioschka



Native packer

Matrioschka




```
[/data/app/com.alcineo.bonappetit-oAqKtKb4R8OsKum8Dl7Thw==/base.apk +0xb61b4]   mmap (  start_addr = 0x0 , size = 0x40
[0x77d0675000 +0x4b0]   mmap (  start_addr = 0x772c1be000 , size = 0x1000 , prot = PROT_READ | PROT_WRITE | PROT_EXEC
[0x772c1be000 +0x2bc]   mmap (  start_addr = 0x77fd4ad000 , size = 0x1000 , prot = PROT_READ | PROT_WRITE | PROT_EXEC
[0x772c1be000 +0x2bc]   mmap (  start_addr = 0x7829640000 , size = 0x1000 , prot = PROT_READ | PROT_WRITE | PROT_EXEC
[0x772c1be000 +0x2bc]   mmap (  start_addr = 0x78a0932000 , size = 0x1000 , prot = PROT_READ | PROT_WRITE | PROT_EXEC
[0x772c1be000 +0x2bc]   mmap (  start_addr = 0x78efc19000 , size = 0x1000 , prot = PROT_READ | PROT_WRITE | PROT_EXEC
[0x772c1be000 +0x2bc]   mmap (  start_addr = 0x785451b000 , size = 0x1000 , prot = PROT_READ | PROT_WRITE | PROT_EXEC
[0x772c1be000 +0x2bc]   mmap (  start_addr = 0x7818ecd000 , size = 0x1000 , prot = PROT_READ | PROT_WRITE | PROT_EXEC
[0x772c1be000 +0x2bc]   mmap (  start_addr = 0x783c358000 , size = 0x1000 , prot = PROT_READ | PROT_WRITE | PROT_EXEC
[0x7855975000 +0xe0]   openat (  dfd = AT_FDCWD , filename = /proc/self/status , int flags = 0x0 , mode = 0x0 )    >
[0x7855975000 +0xe0]   read (  fd = 37  /proc/self/status    , buf = 0x7ff14c6be0 , count = 0xfff  )    > (len=0x414)
[0x7855975000 +0xe0]   close (  fd = 37  /proc/self/status    )    > 0x0
```

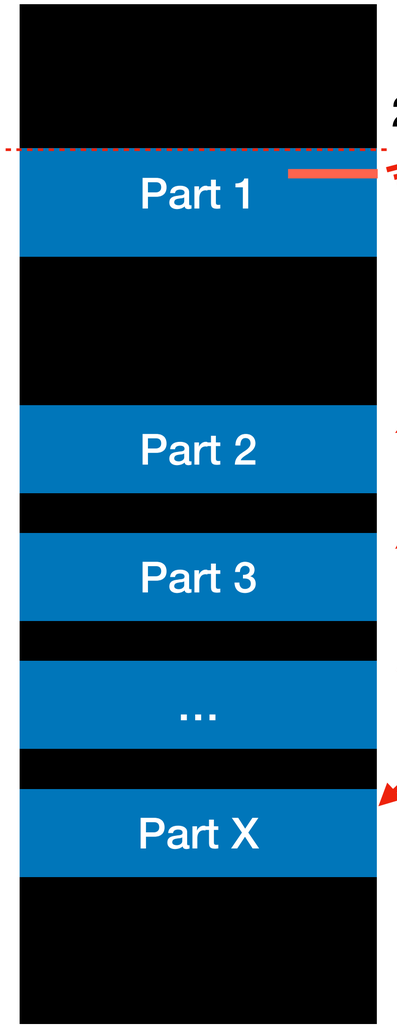
Interruptor output

openat(/proc/self/status)

Common ftrace detection (anti-debug)

Native packer

Matrioschka



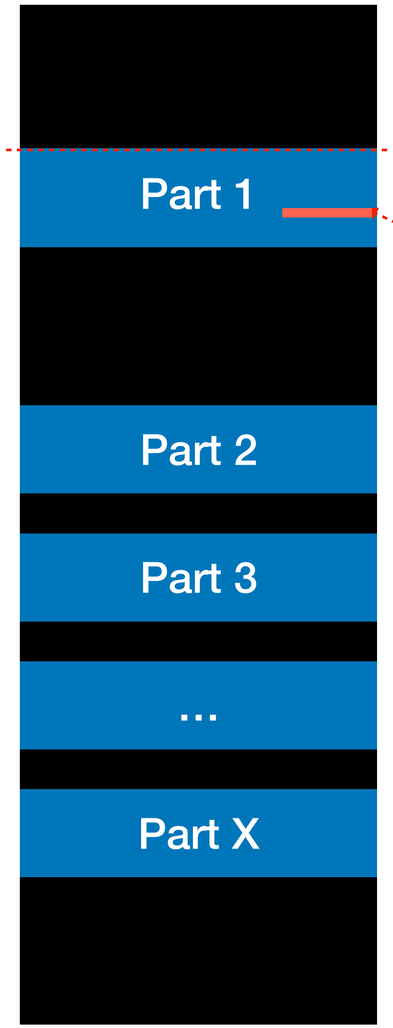
2/ mmap RWX

```
const P = KAPI.CONST.PROT_;
const mmaps = [];
Interruptor.newAgentTracer({
    followThread: false,
    svc: {
        mmap: {
            onLeave: function(ctx){
                const len = ctx.x1.toUInt32();
                if(ctx.x2.toUInt32() == (P.PROT_READ | P.PROT_WRITE | P.PROT_EXEC)
                    && (len==0x1000  ||  len==0x4000)){
                    mmaps.push({ addr:ctx.x0, len:len });
                }
            }
        },

    }
});
```

Interruptor script

Native packer

Matrioschka

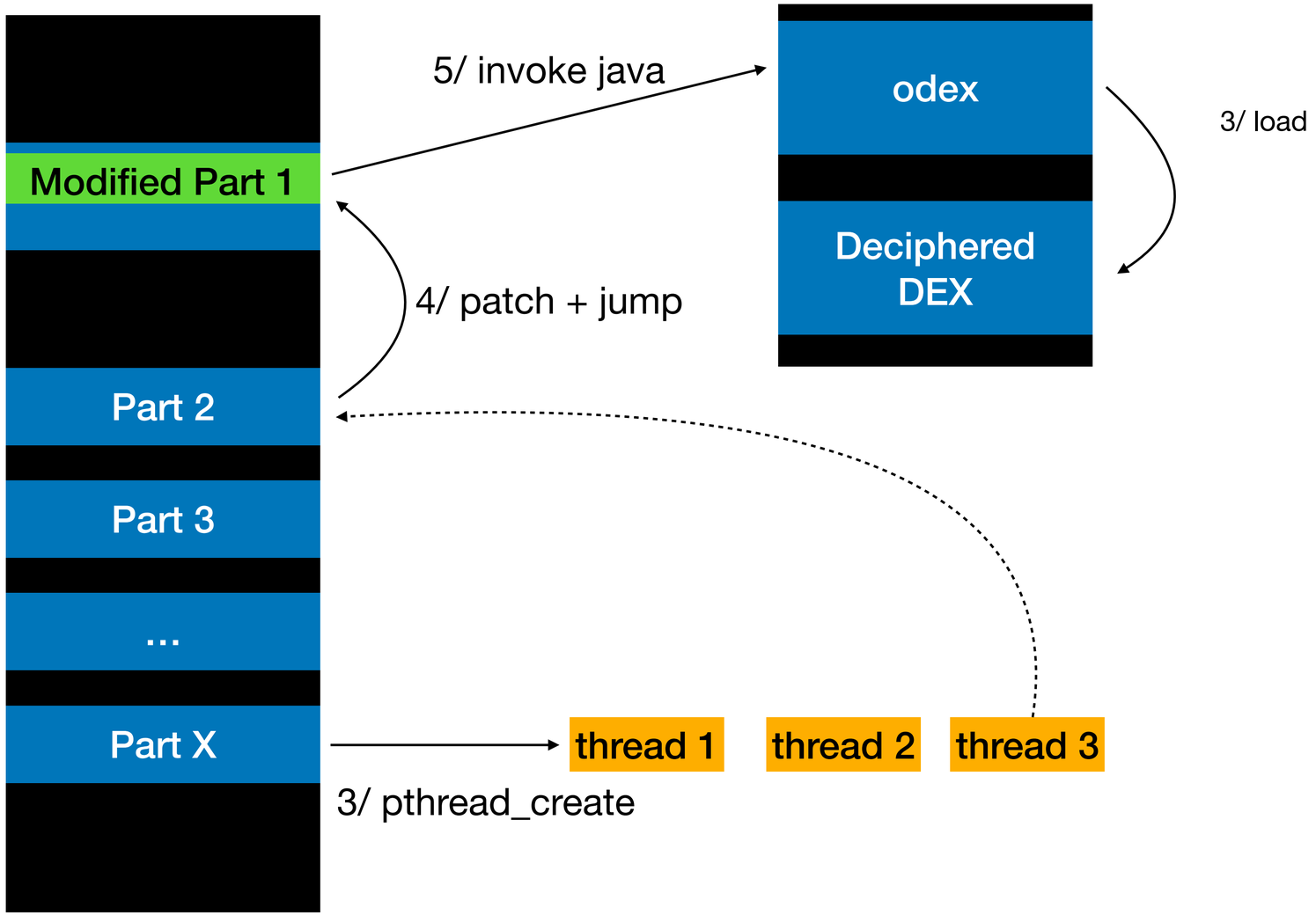



```
const P = KAPI.CONST.PROT_;
const mmaps = [];
Interruptor.newAgentTracer({
    followThread: false,
    svc: {
        mmap: {
            onLeave: function(ctx){
                const len = ctx.x1.toUInt32();
                if(ctx.x2.toUInt32() == (P.PROT_READ | P.PROT_WRITE | P.PROT_EXEC)
                    && (len==0x1000  ||  len==0x4000)){
                    mmaps.push({ addr:ctx.x0, len:len });
                }
            }
        },
        openat: {
            onEnter: function(ctx){
                const fn = ctx.x1.readCString();
                if(fn.indexOf("/proc")==0  && mmaps.length > 0){
                    mmaps.map( x => {
                        const f = new File("/data/data/"+PKG+"/dump_"+x.addr.toString(16)+"_"+x.len+".bin","wb");
                        f.write(x.addr.readByteArray(x.len));
                        f.flush();
                        f.close();
                    });
                    mmaps = []
                    console.log("Memory segments have been dumped.");
                }
            }
        }
    }
});
```

Interruptor script

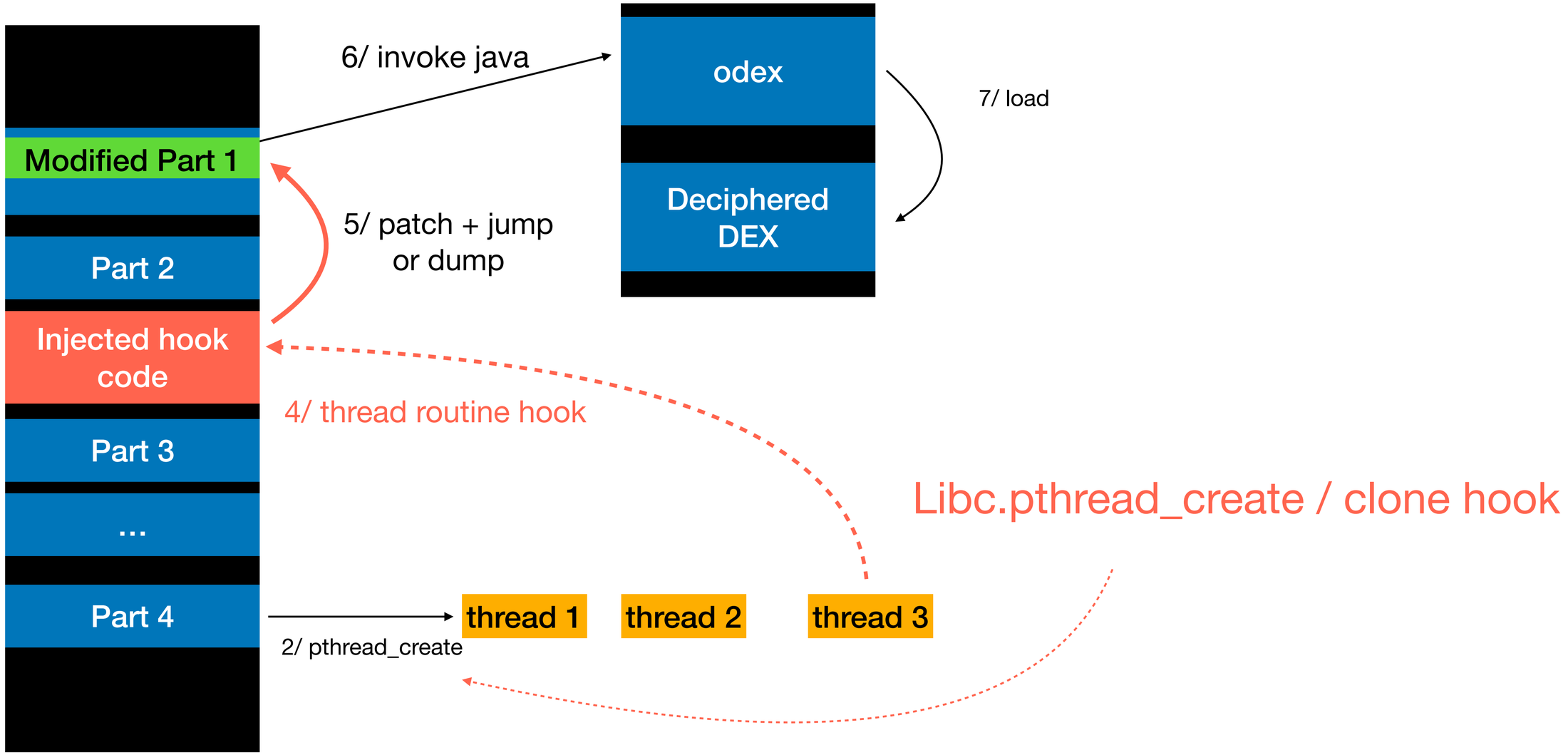
Native packer

Matrioschka



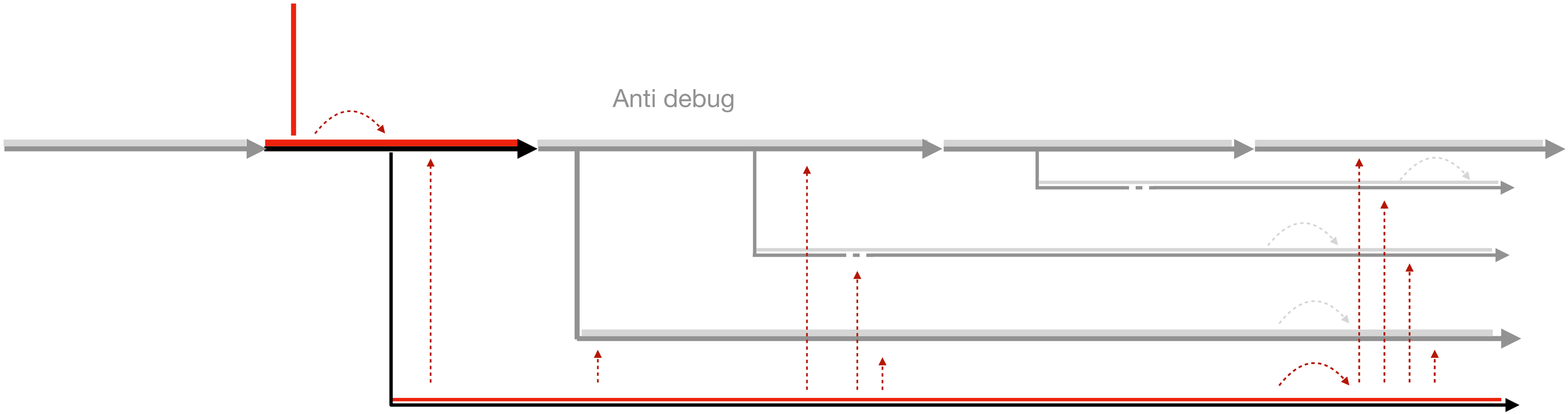
Native packer

Matrioschka



[anti-] Hook

Hook Detection

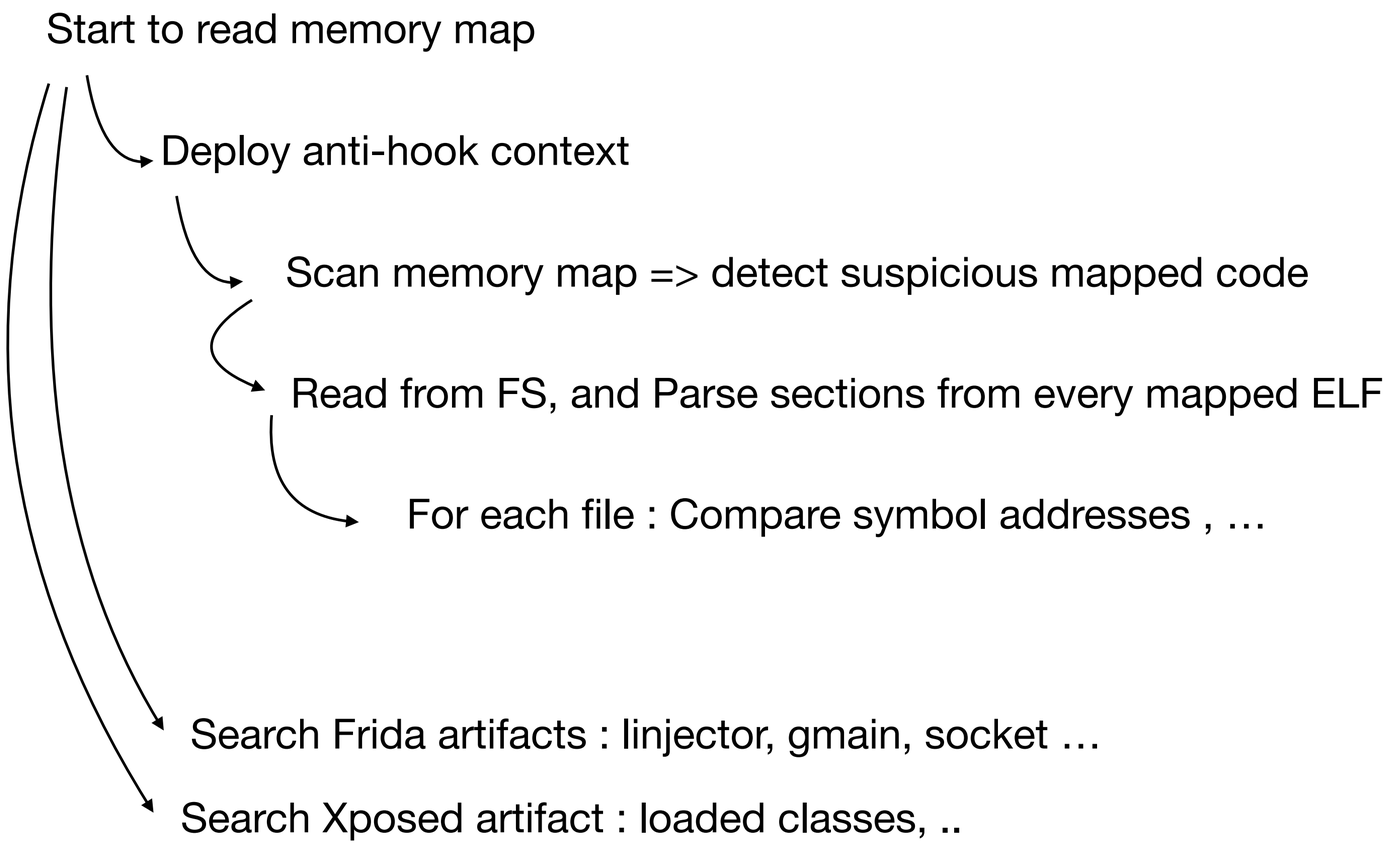


thread

fork

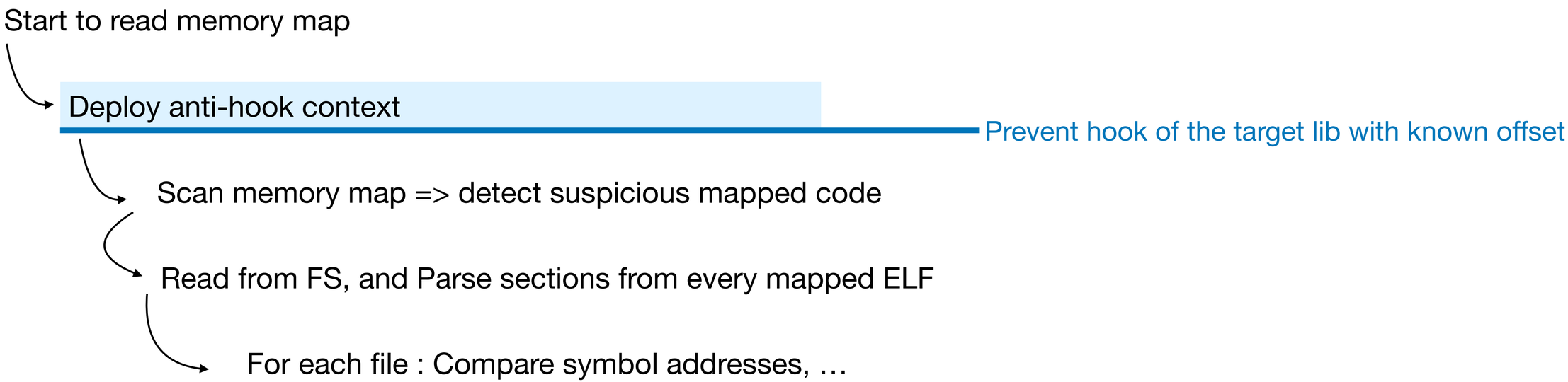
Anti Hook

Roadmap



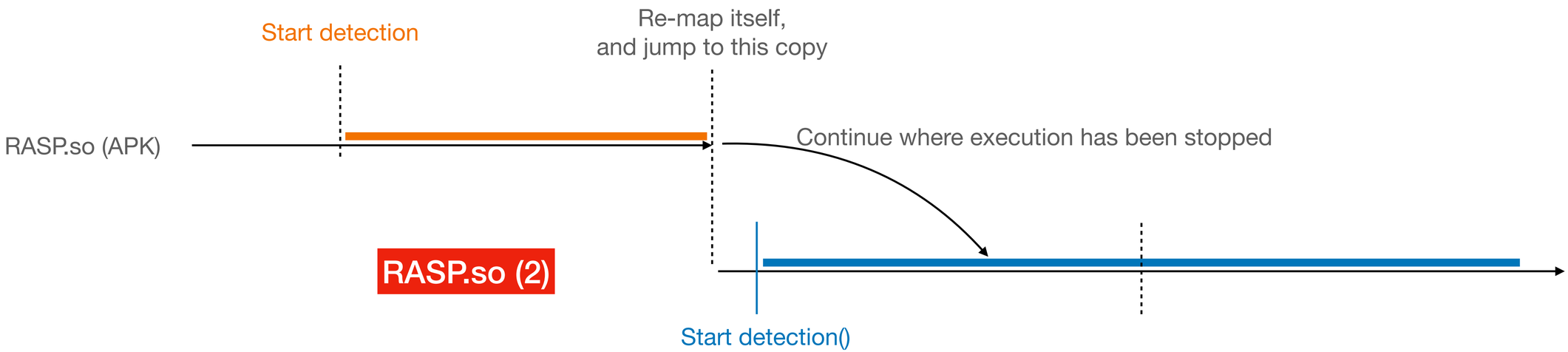
Anti Hook

Roadmap



Anti Hook

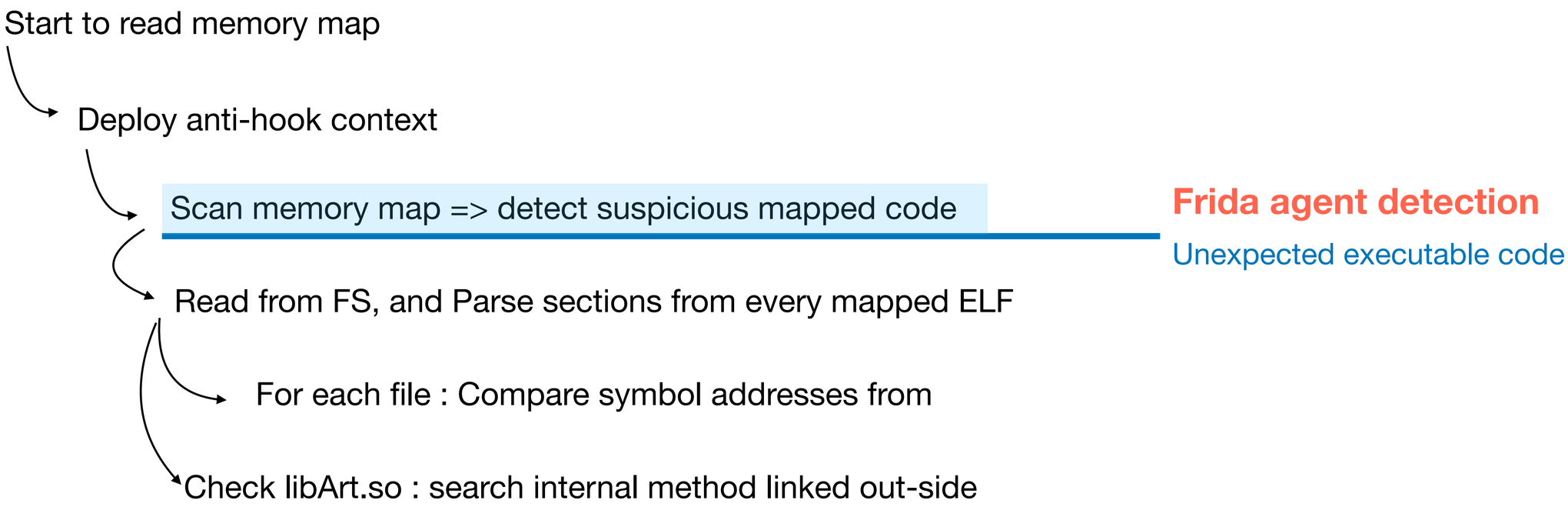
Deploy anti-hook context



Hooking require to know the absolute offset
(offset of RASP.so(2) + relative offset of hooked method)

Anti Hook

Roadmap



Anti Hook

Detect suspicious executable code

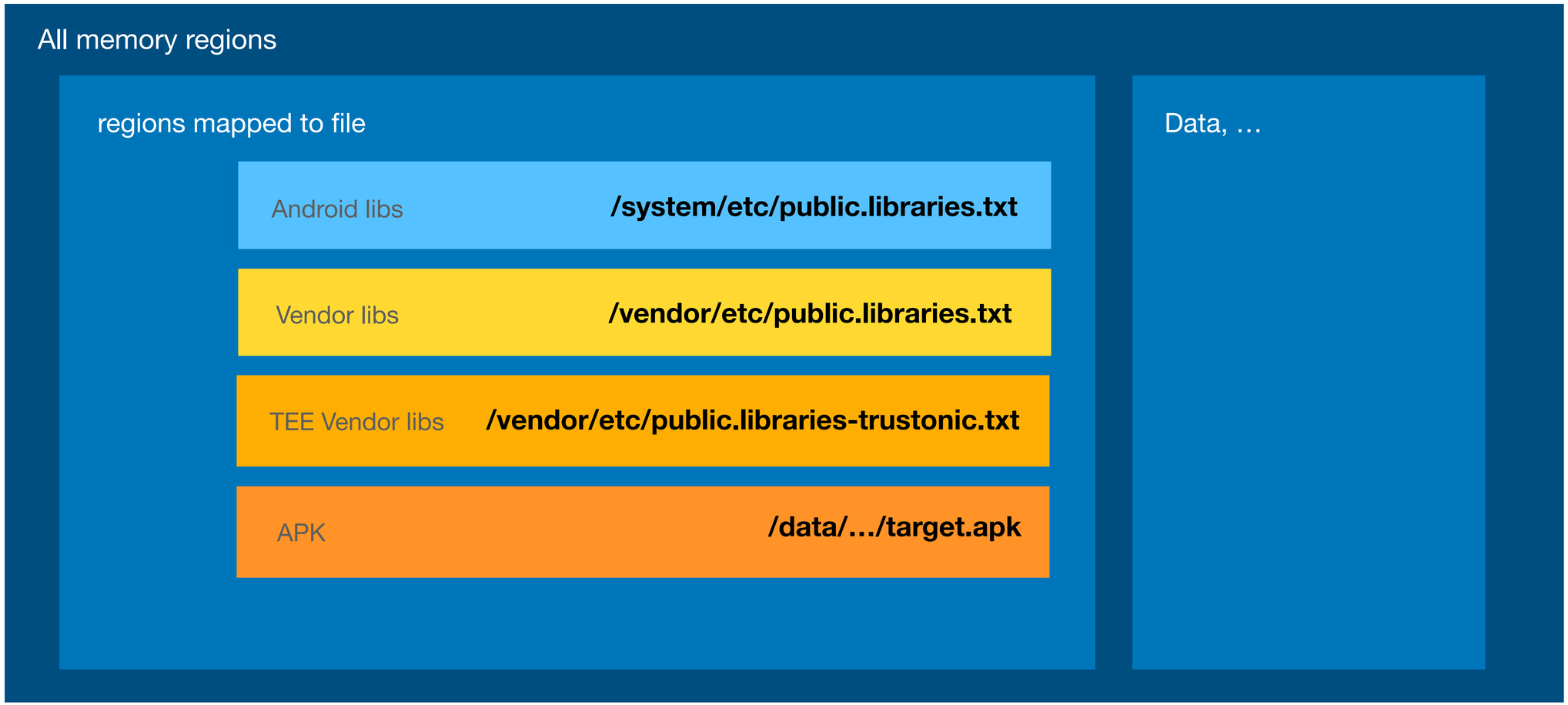
All memory regions

Anti Hook

Detect suspicious executable code

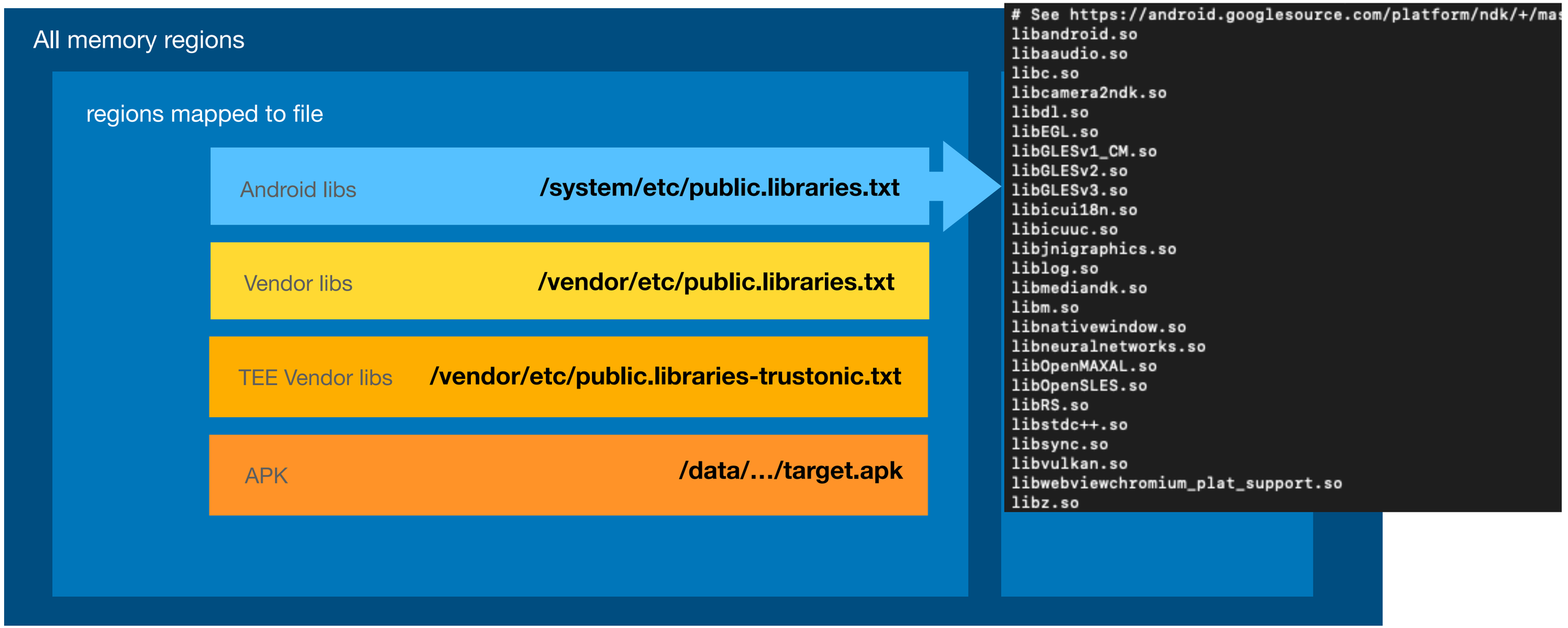
Anti Hook

Detect suspicious executable code



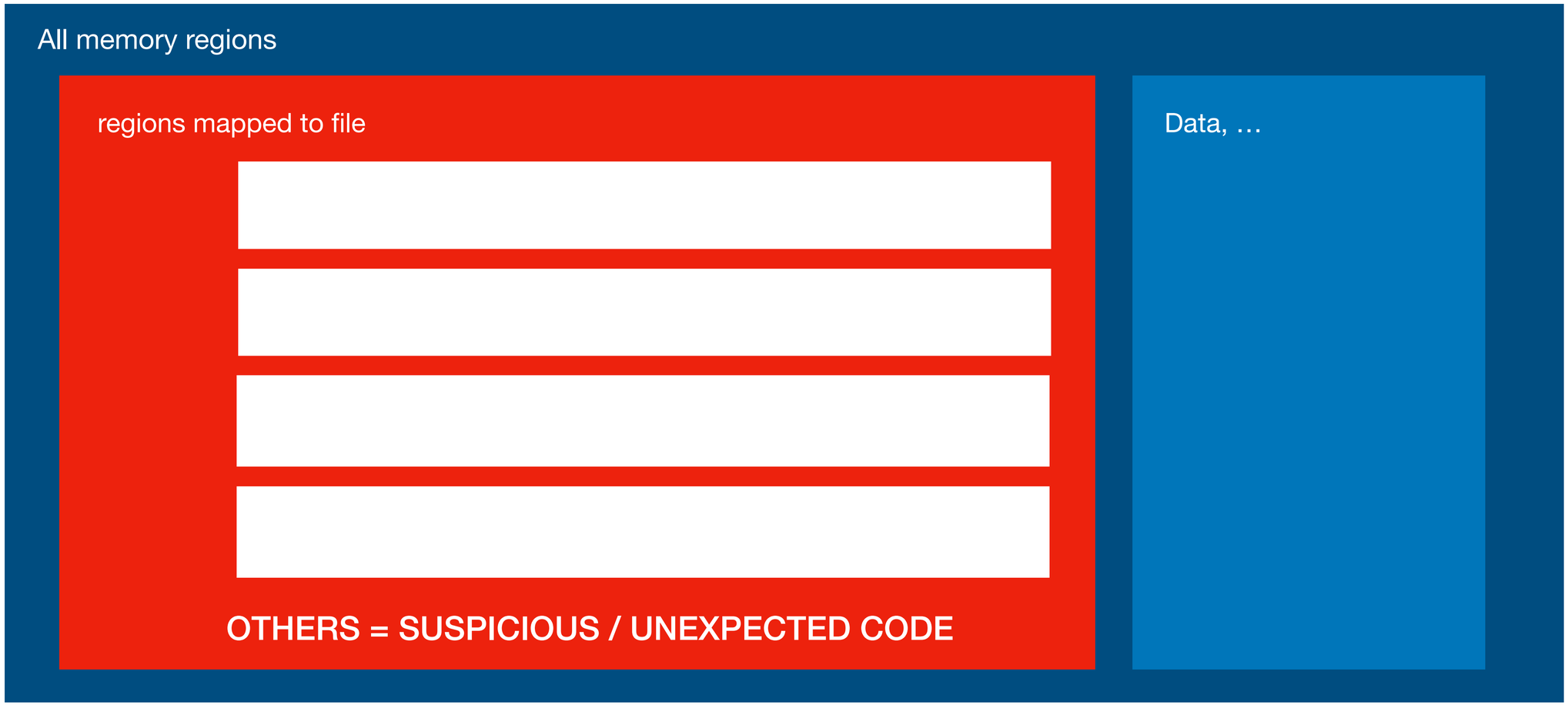
Anti Hook

Detect suspicious executable code



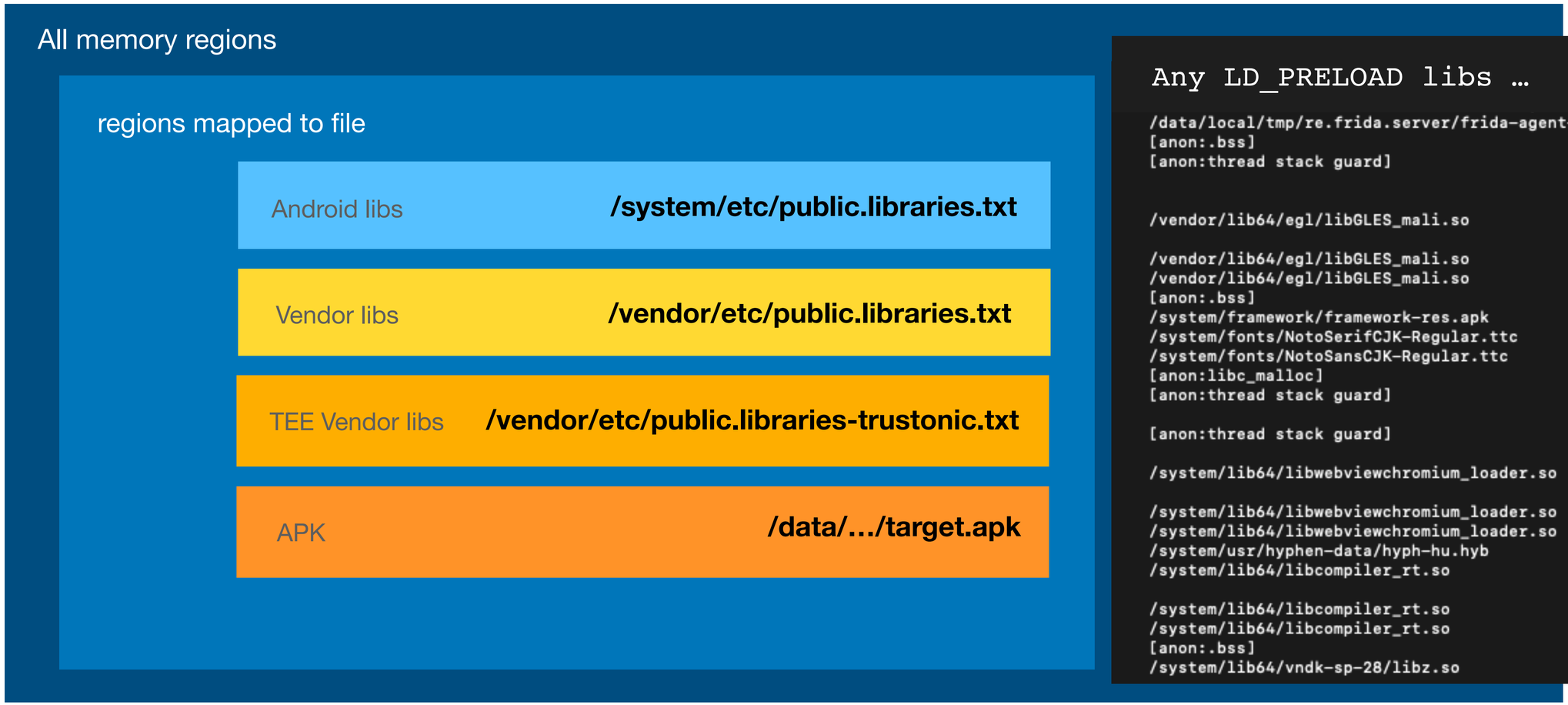
Anti Hook

Detect suspicious executable code



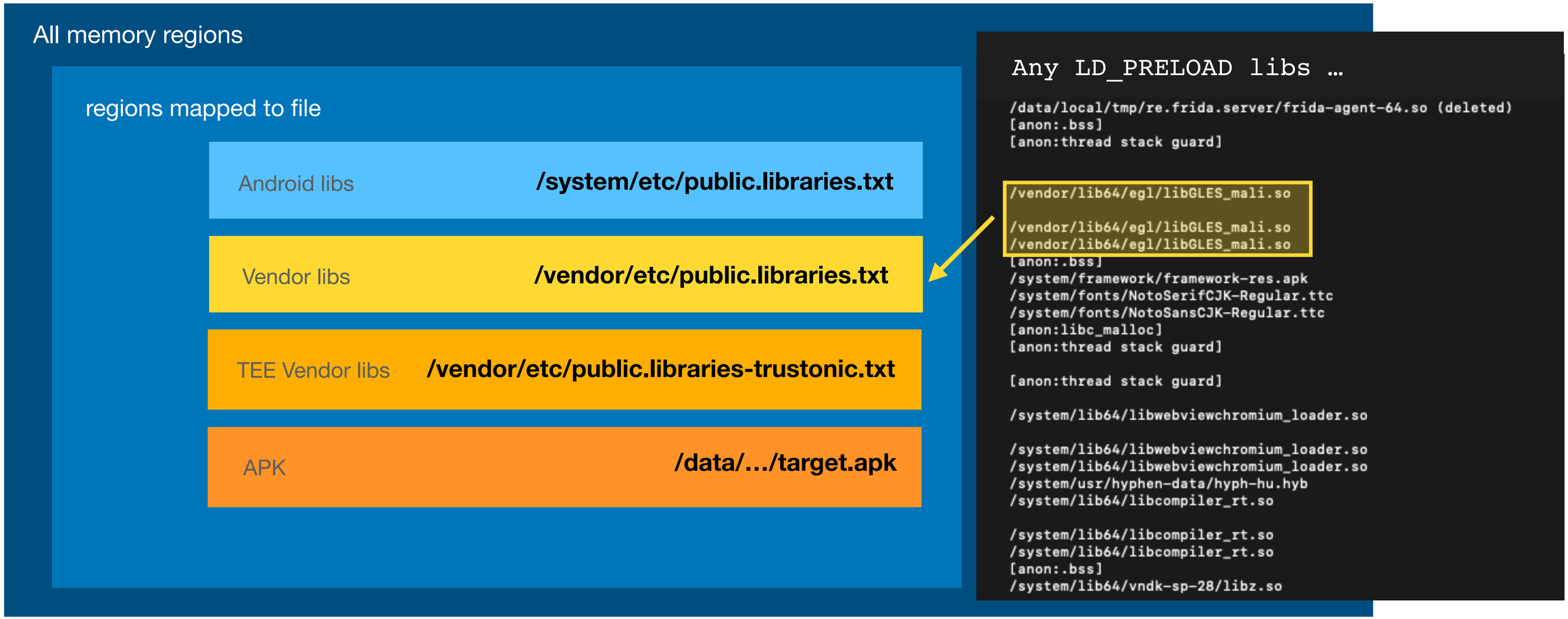
Anti Hook

Detect suspicious executable code



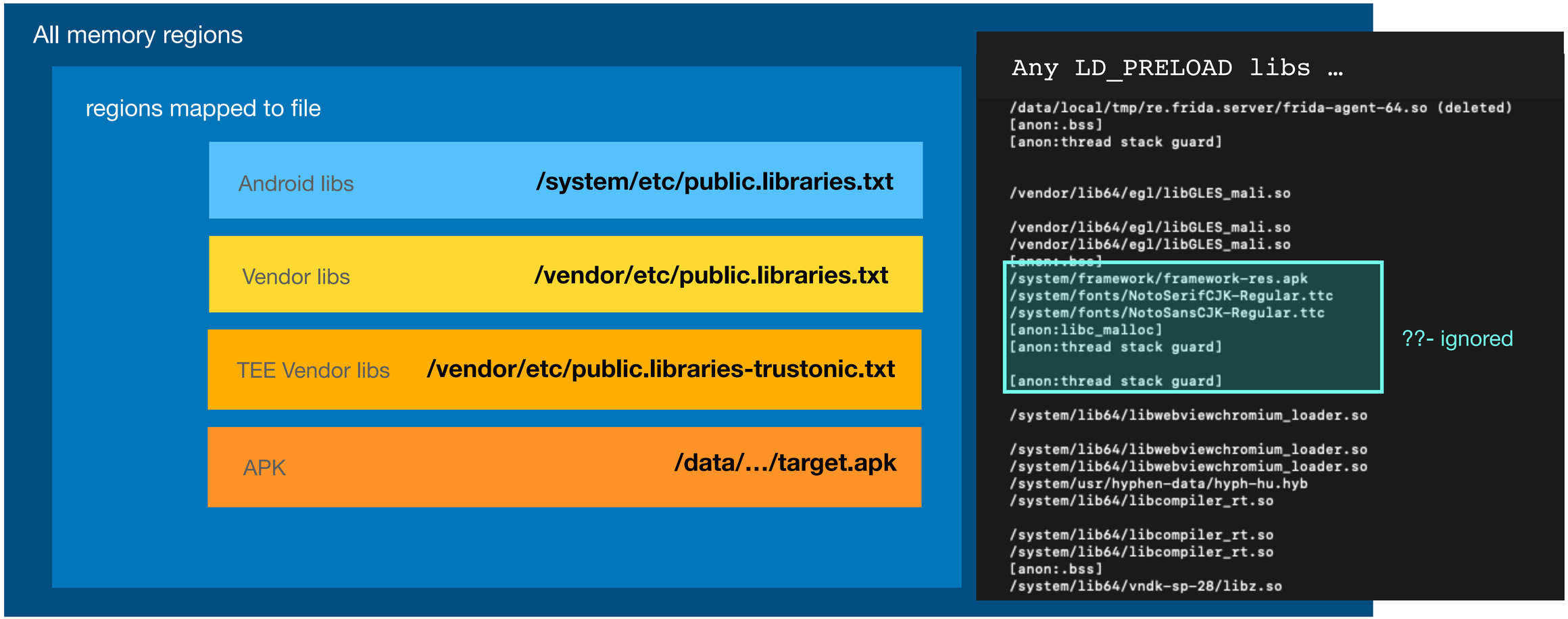
Anti Hook

Detect suspicious executable code



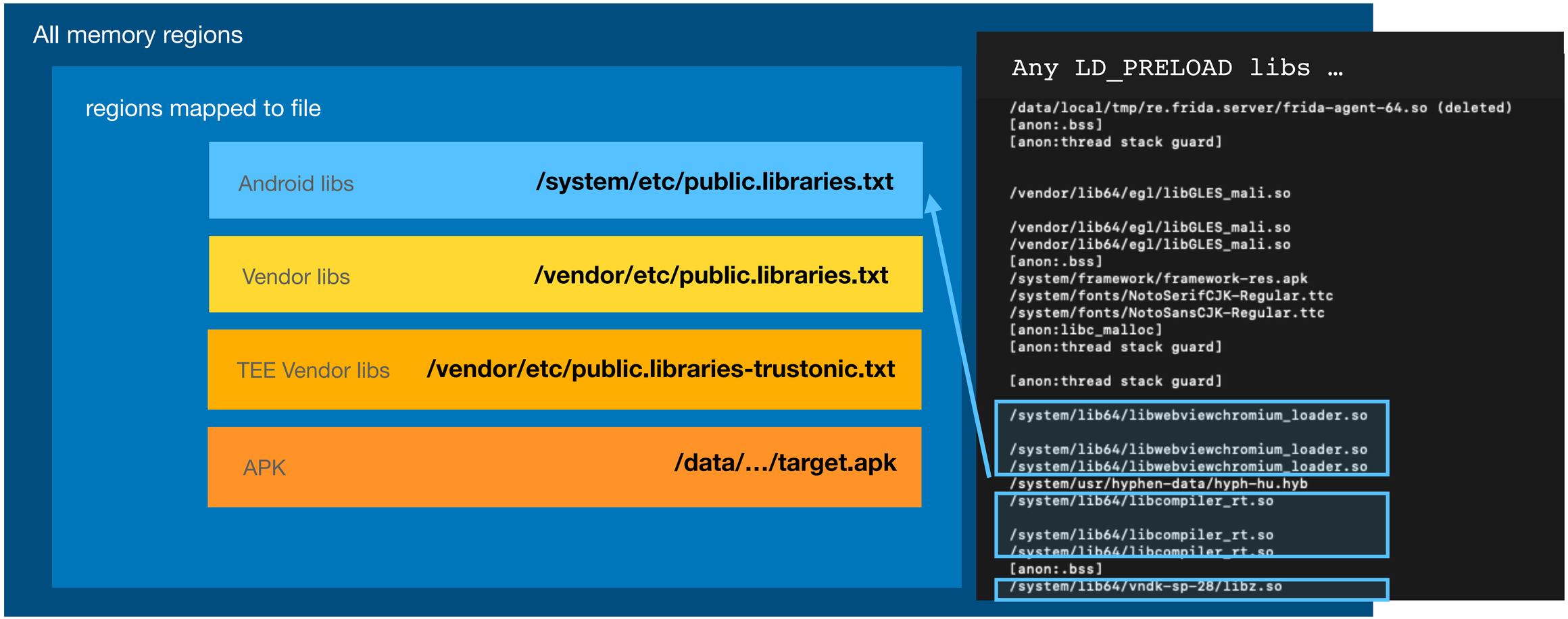
Anti Hook

Detect suspicious executable code



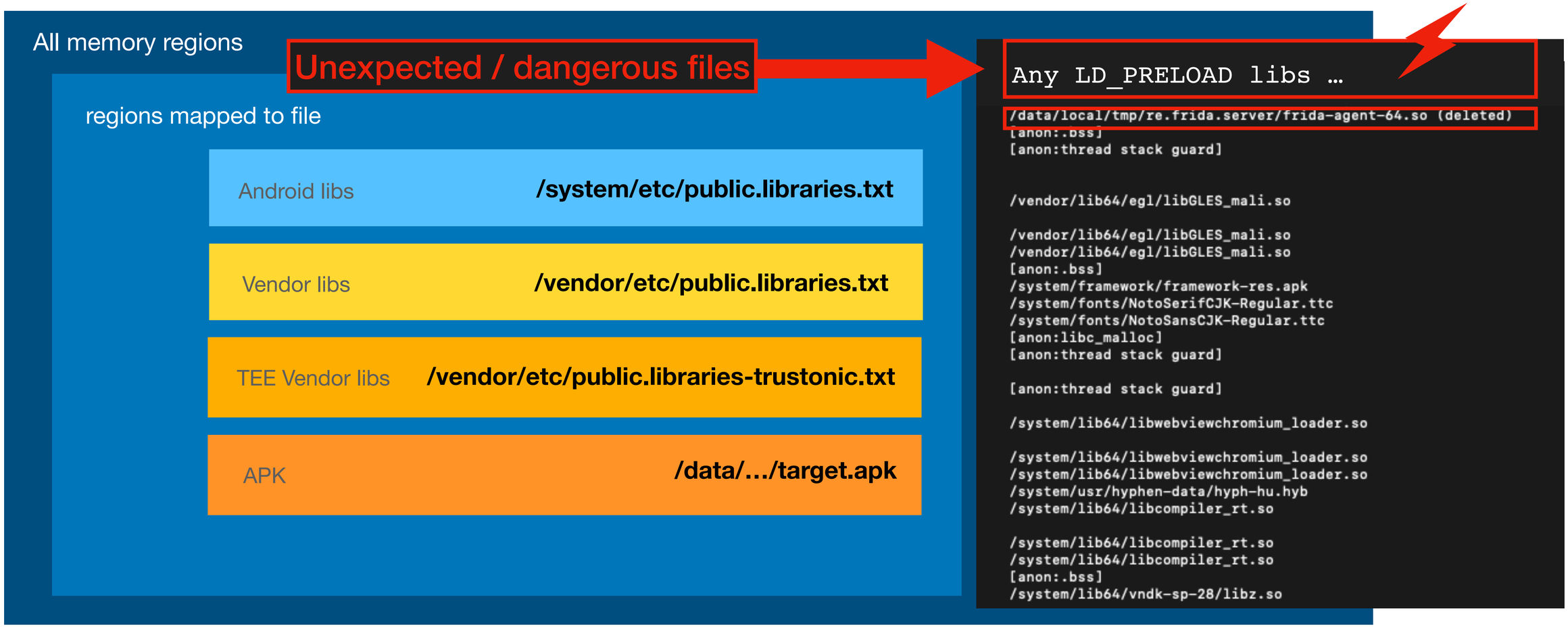
Anti Hook

Detect suspicious executable code



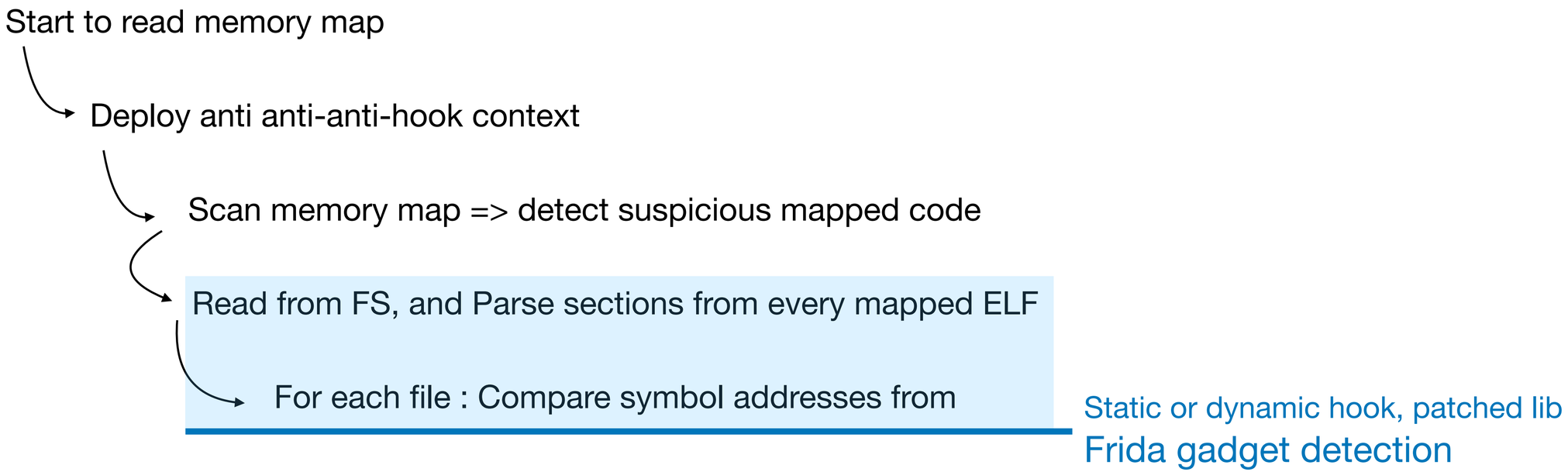
Anti Hook

Detect suspicious executable code



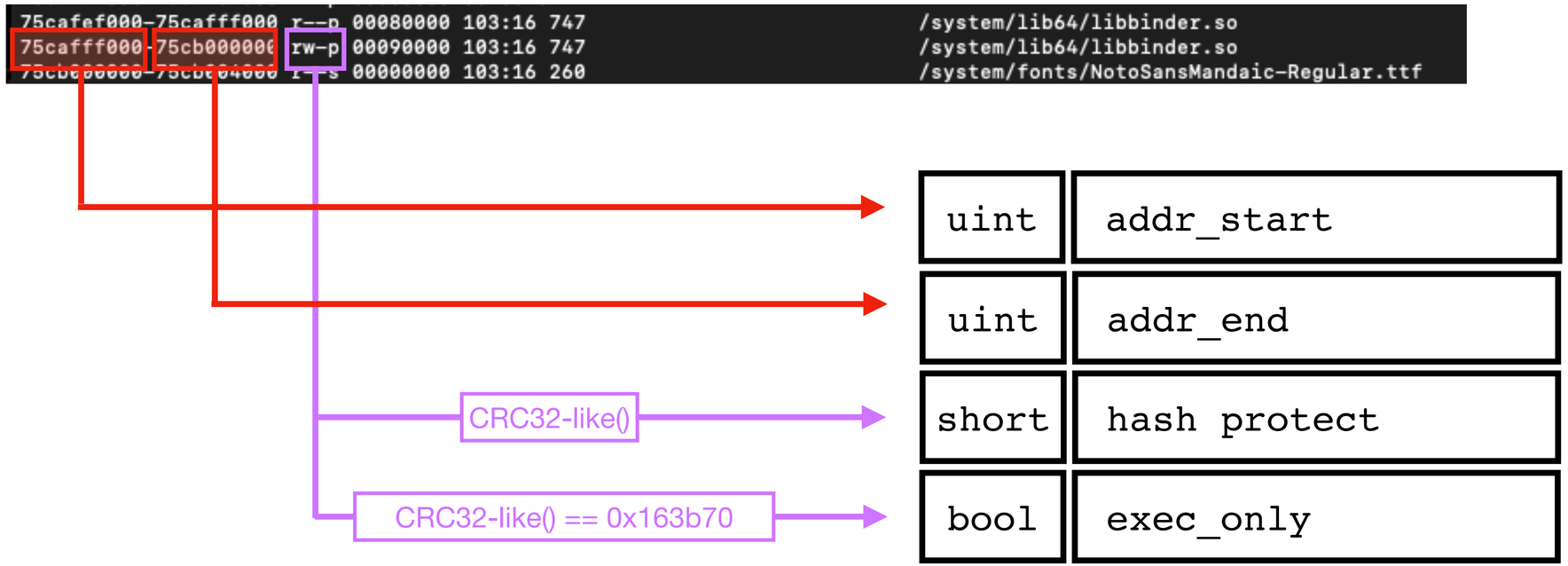
Anti Hook

Roadmap



Anti Hook

Memory map parsing



Anti Hook

Memory map parsing

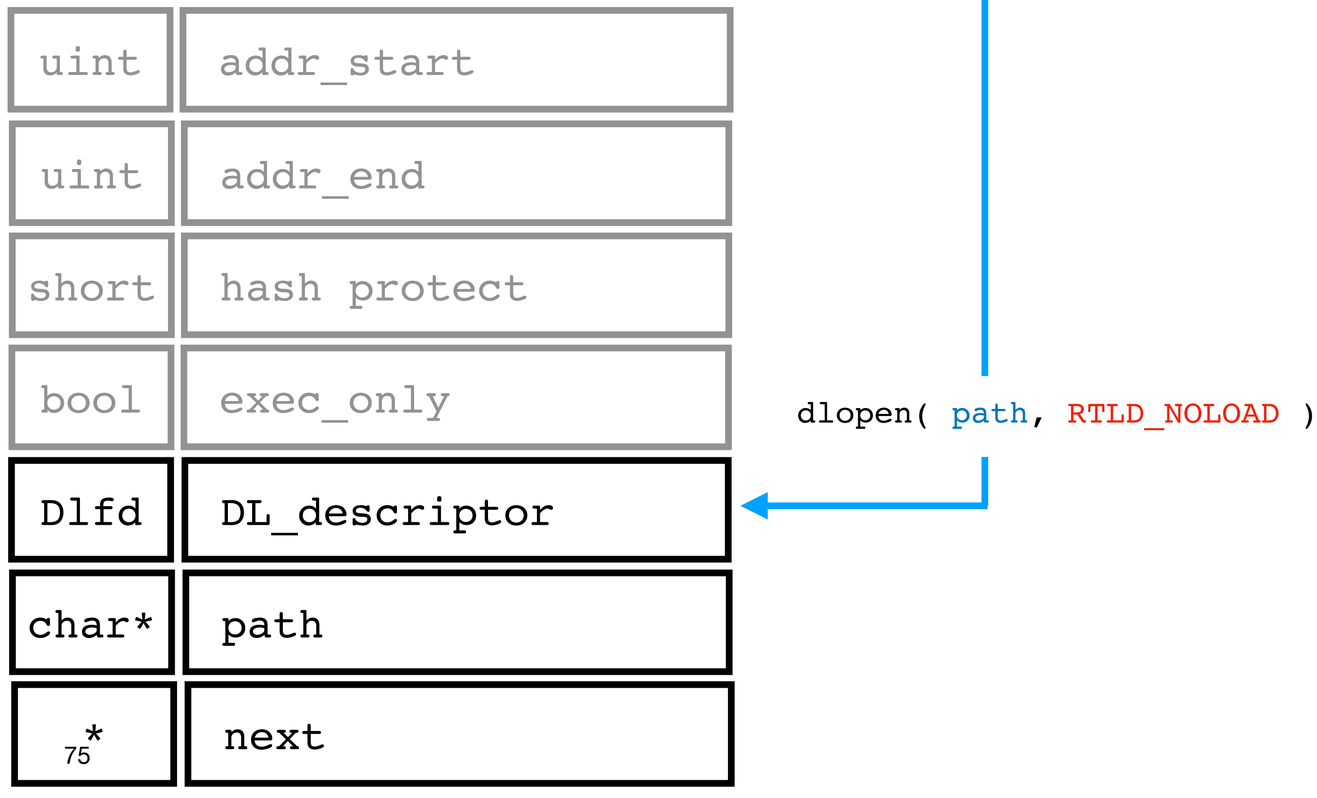
```
      if (iVar3 != 0x360f) {
        pcVar6 = basename((char *)path);
        uVar4 = hash_2(pcVar6);
        if ((uVar4 != 0xdea6a0d2) && (uVar4 != 0x9a834d8e)) goto not_linker64;
      }
      v = *(int *)ULONG_0011eff8;
      if ((28 < v) || ((hash_prot == 0xf70 || (hash_prot == 0xf2d)))) {
        if (list_mods == (astruct_4 *)0x0) {
append_to_chained_list:
          lib_info = (astruct_4 *)malloc(0x30);
          lib_info->addr_start = range_start_addr;
          lib_info->addr_end = range_end_addr;
          lib_info->size = hash_size;
          lib_info->exec_only = 0;
          lib_info->dl_fd = 0;
          if (v < 24) {
builtin_libraries:
            fd = dlopen(path, RTLD_NOLOAD);
            lib_info->dl_fd = fd;
          }
          else {
```

Detect if the lib is already loaded

Anti Hook

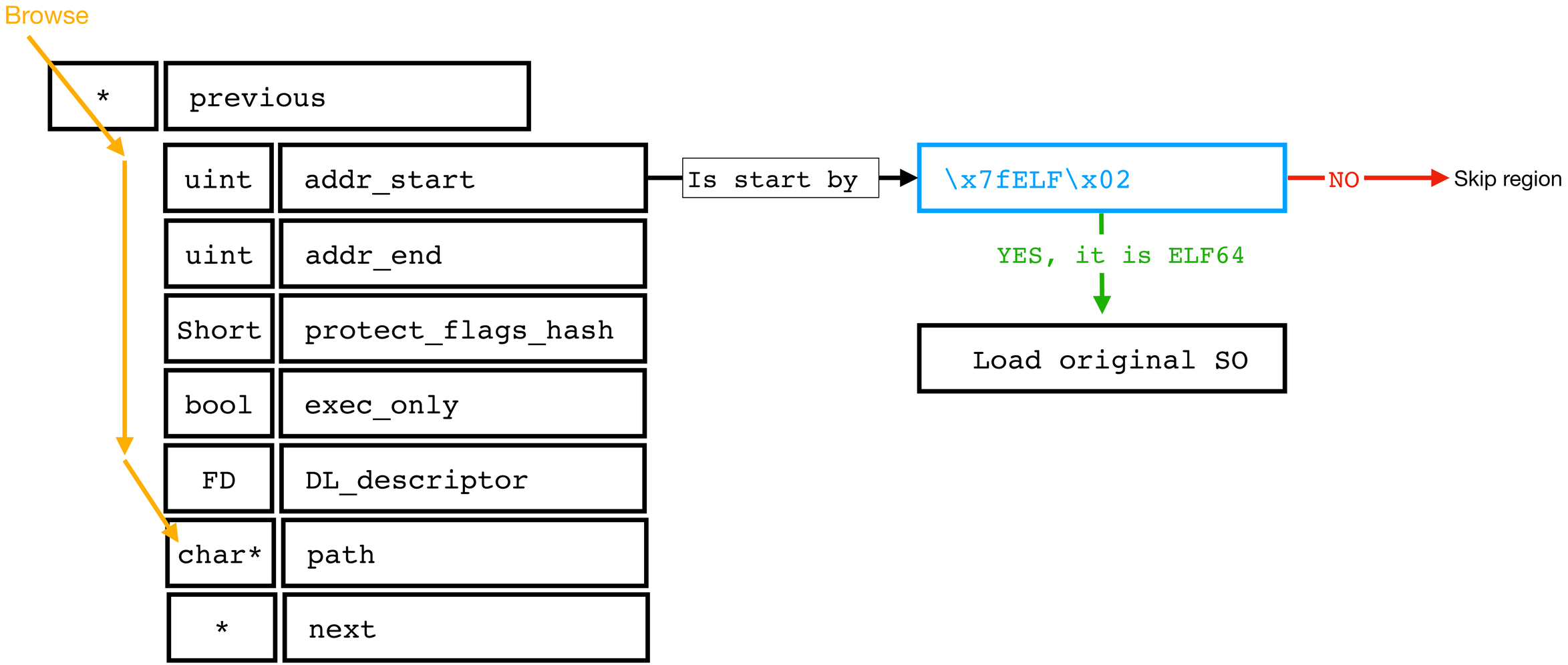
Memory map parsing

```
75cafef000-75cafff000 r--p 00080000 103:16 747                  /system/lib64/libbinder.so
75cafff000-75cb000000 rw-p 00090000 103:16 747                  /system/lib64/libbinder.so
75cb000000-75cb004000 r--s 00000000 103:16 260                  /system/fonts/NotoSansMandaic-Regular.ttf
```



Anti Hook

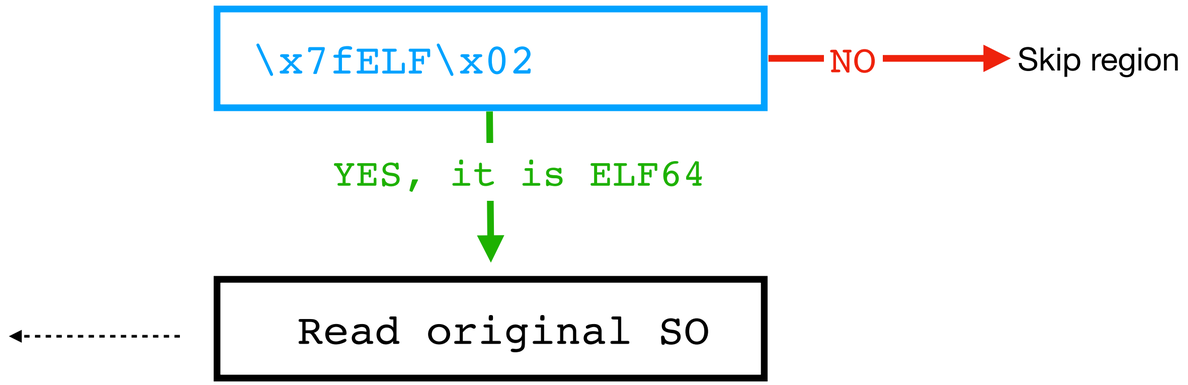
Memory map parsing



Anti Hook

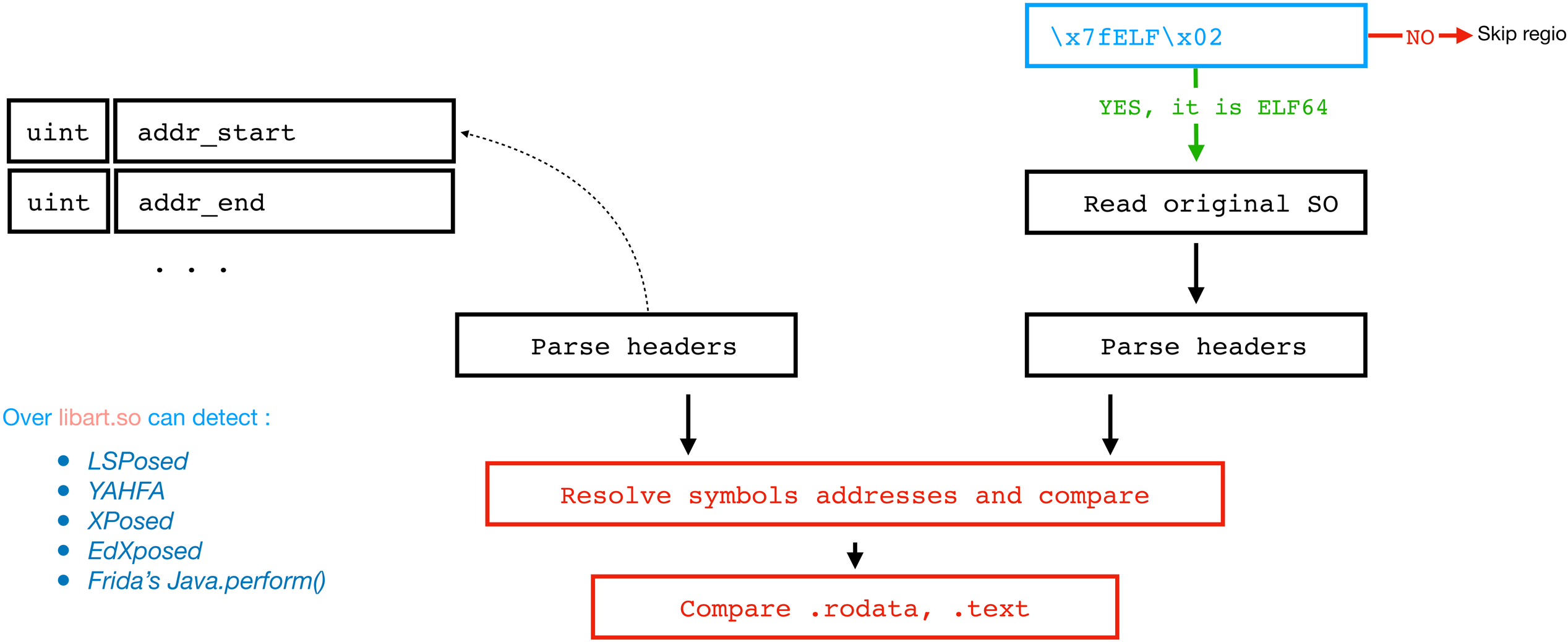
Memory map parsing

```
fd = fopen_with_svc(binInfo->path,0);
if (fd != -1) {
  size = svc_lseek(fd,0,2);
  elf_file_addr = (void *)svc_mmap(0,size,1,1,fd,0);
  if (elf_file_addr != (void *)0xffffffffffffffff) {
    iVar1 = check_symbols(elf_file_addr,binInfo,(ulong)nonce,result);
    svc_munmap(elf_file_addr,size);
    if (iVar1 != 0) {
      fclose_svc(fd);
      return iVar1;
    }
  }
  fclose_svc(fd);
}
return 0;
```



Anti Hook

Memory map parsing

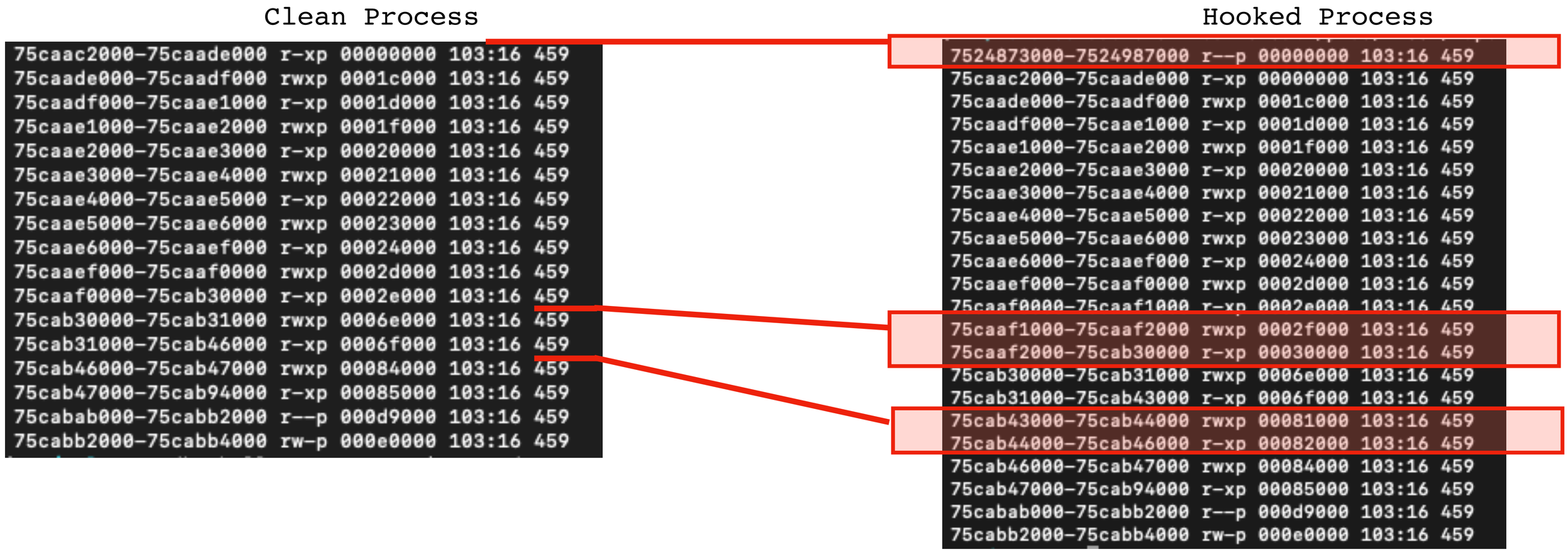


Over libart.so can detect :

- *LSPosed*
- *YAHFA*
- *XPosed*
- *EdXposed*
- *Frida's Java.perform()*

Anti Hook

Regions mapped to /system/lib64/libc.so



Anti Hook

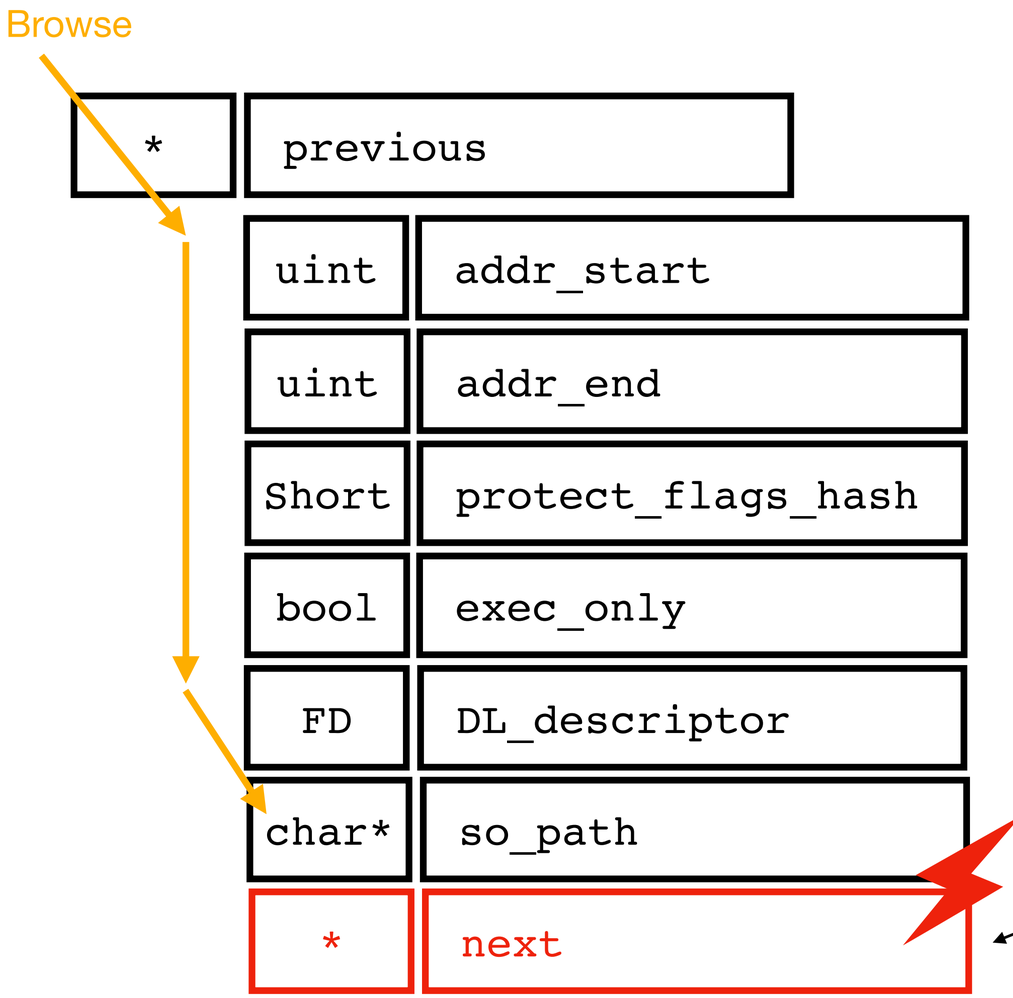
Bypasses

1/ Patch linked list to stop the analysis of regions

2/ Lift output buffer of read() sys call when /proc/*/maps is read

Anti Hook

Case 1 : Patch linked list to stop region analysis



It works finely but is too much specific to RASP internals

Hook `Parse_header()` and Patch « `next` » with NULL

Anti Hook

Case 2 : lifting read output of /proc/*/maps

Easy case

Each hook call
is independent,
Bypass is local

```
75ccef8000-75ccf18000 r--s 00000000 00:11 14752                          /dev/__properties__/pr
75ccf18000-75ccf1a000 rw-p 00000000 00:00 0                              [anon:System property
75ccf1a000-75ccf32000 r-xp 00000000 103:16 2175                          /system/bin/linker64
75ccf32000-75ccf33000 rwxp 00018000 103:16 2175                          /system/bin/linker64
75ccf33000-75ccf46000 r-xp 00019000 103:16 2175                          /system/bin/linker64
75ccf46000-75ccf47000 rwxp 0002c000 103:16 2175                          /system/bin/linker64
75ccf47000-75cd041000 r-xp 0002d000 103:16 2175                          /system/bin/linker64
75cd041000-75cd04b000 r--s 00000000 00:11 14600                          /dev/__properties__/pr
75cd04b000-75cd04c000 r--p 00000000 00:00 0                              [anon:linker_alloc]
```

Anti Hook

Case 2 : lifting read output of /proc/*/maps

Easy case

Each hook call is independent, Bypass is local

```
75ccef8000-75ccf18000 r--s 00000000 00:11 14752                  /dev/__properties__/pr
75ccf18000-75ccf1a000 rw-p 00000000 00:00 0                      [anon:System property
75ccf1a000-75ccf32000 r-xp 00000000 103:16 2175                  /system/bin/linker64
75ccf32000-75ccf33000 rwxp 00018000 103:16 2175                  /system/bin/linker64
75ccf33000-75ccf46000 r-xp 00019000 103:16 2175                  /system/bin/linker64
75ccf46000-75ccf47000 rwxp 0002c000 103:16 2175                  /system/bin/linker64
75ccf47000-75cd041000 r-xp 0002d000 103:16 2175                  /system/bin/linker64
75cd041000-75cd04b000 r--s 00000000 00:11 14600                  /dev/__properties__/pr
75cd04b000-75cd04c000 r--p 00000000 00:00 0                      [anon:linker_alloc]
```

Read / process each line

Hardened case

Hook holds state and change behavior of next hook call

```
75ccef8000-75ccf18000 r--s 00000000 00:11 14752                  /dev/__properties__/pr
75ccf18000-75ccf1a000 rw-p 00000000 00:00 0                      [anon:System property
75ccf1a000-75ccf32000 r-xp 00000000 103:16 2175                  /system/bin/linker64
75ccf32000-75ccf33000 rwxp 00018000 103:16 2175                  /system/bin/linker64
75ccf33000-75ccf46000 r-xp 00019000 103:16 2175                  /system/bin/linker64
75ccf46000-75ccf47000 rwxp 0002c000 103:16 2175                  /system/bin/linker64
75ccf47000-75cd041000 r-xp 0002d000 103:16 2175                  /system/bin/linker64
75cd041000-75cd04b000 r--s 00000000 00:11 14600                  /dev/__properties__/pr
75cd04b000-75cd04c000 r--p 00000000 00:00 0                      [anon:linker_alloc]
```

Read char by char

```
75ccef8000-75ccf18000 r--s 00000000 00:11 14752                  /dev/__properties__/pr
75ccf18000-75ccf1a000 rw-p 00000000 00:00 0                      [anon:System property
75ccf1a000-75ccf32000 r-xp 00000000 103:16 2175                  /system/bin/linker64
75ccf32000-75ccf33000 rwxp 00018000 103:16 2175                  /system/bin/linker64
75ccf33000-75ccf46000 r-xp 00019000 103:16 2175                  /system/bin/linker64
75ccf46000-75ccf47000 rwxp 0002c000 103:16 2175                  /system/bin/linker64
75ccf47000-75cd041000 r-xp 0002d000 103:16 2175                  /system/bin/linker64
75cd041000-75cd04b000 r--s 00000000 00:11 14600                  /dev/__properties__/pr
75cd04b000-75cd04c000 r--p 00000000 00:00 0                      [anon:linker_alloc]
```

Read byte sequences with random length

Anti Hook

Case 2 : lifting read output of /proc/*/maps (easy case)

```
read: {
    onEnter: function(ctx){
        const f = ctx.dxcFD[ctx.x0.toInt32()];

        if( /^\/proc\/.+\/maps$/g.exec(f)){
            ctx.maps = true;
        }
    },
    onLeave: function(ctx){

        if(ctx.maps){
           ctx.maps = false;

           SCAN & PATCH read sys call OUTPUT BUFFER (referenced by x1)
        }
    }
}
```

Hook of read sys call (x8 = 0x3f ; …. ; svc #0) with Interruptor

Anti Hook

Case 2 : lifting read output of /proc/*/maps (easy case)

```
let res = null;
res = Memory.scanSync(ctx.x1, ctx.x2.toInt32(), RWX_PATTERN);
if(res.length > 0){
    res.map( m => m.address.writeByteArray([0x72,0x2D,0x78]));
    console.warn("replace 'rwx' by 'r-x' from resulting buffer");
}
```

Replace « rwx » protection flags by « r-x »

```
res = Memory.scanSync(ctx.x1, ctx.x2.toInt32(), LIBC_PATTERN);
if(res.length > 0){
     res.map( m => m.address.writeByteArray([0x6c,0x69,0x62,0x71,0x2E]));
     console.warn("replace 'libc.' by 'libz.' pattern from resulting buffer");
     return ;
}
```

Patch filename « libc. » with « libz. »

Anti Frida

Frida injector detection

```
 PID=12435 TID=[redacted]:12536  libc > vfork > [PID=12435] [Parent TID=12536] fork
 PID=12435 TID=12535  libc > __open_2 > /dev/ashmem 0x53
File (1) : new : /data/app/com.google.android.gms-vPxYPvx1h7yBxA_QtcO2ww==/lib/arm64
 PID=12435 TID=[redacted]12536  libc > fork > [PID=12435] [Parent TID=12536] fork
File (1) : new : /data/app/com.google.android.gms-vPxYPvx1h7yBxA_QtcO2ww==/base.apk!/lib/arm64-v8a
File (1) : new : /data/app/com.google.android.gms-vPxYPvx1h7yBxA_QtcO2ww==/split_config.fr.apk!/lib/arm64-v8a
 String.contains( /data/app/com.google.android.gms-vPxYPvx1h7yBxA_QtcO2ww==/lib/arm64, !/ ) = false :
 PID=12435 TID=12535  libc > stat64(/data/app/com.google.android.gms-vPxYPvx1h7yBxA_QtcO2ww==/lib/arm64, 0x7a41d5fa28) = 0x0
 String.contains( /data/app/com.google.android.gms-vPxYPvx1h7yBxA_QtcO2ww==/base.apk!/lib/arm64-v8a, !/ ) = true :
File (1) : new : /data/app/com.google.android.gms-vPxYPvx1h7yBxA_QtcO2ww==/base.apk
 PID=12435 TID=12597  libc > open > /data/local/tmp/re.frida.server/linjector-91 0x22
 PID=12435 TID=12597  libc > dlsym (0x835620158e291e43,frida_agent_main) = 0x7c453f35bc
 PID=12435 TID=[redacted]:12536  libc > fork > [PID=12435] [Parent TID=12536] Child PID 12588
 PID=12435 TID=[redacted]:12536  libc > vfork > [PID=12435] [Parent TID=12536] Child PID 12588
 PID=12435 TID=12594  libc > __open_2 > /dev/ashmem 0x51
 String.contains( /data/app/com.google.android.gms-vPxYPvx1h7yBxA_QtcO2ww==/split_config.fr.apk!/lib/arm64-v8a, !/ ) = true :
File (1) : new : /data/app/com.google.android.gms-vPxYPvx1h7yBxA_QtcO2ww==/split_config.fr.apk
```

dlopen / dlsym

Anti Frida

Frida threads detection

```
Name:   plug.androidapp
Name:   Jit thread pool
Name:   Signal Catcher
Name:   ReferenceQueueD
Name:   FinalizerDaemon
Name:   FinalizerWatchd
Name:   HeapTaskDaemon
Name:   gmain
Name:   gdbus
Name:   gum-js-loop
Name:   plug.androidapp
Name:   Binder:31485_1
Name:   Binder:31485_2
```

Frida threads

First line for each /proc/ * /tasks/ * /status

Hook candidates

- `read()` syscall, replace « **gmain** » and « **gum-js-loop** » inside output
- `mmap()` syscall with FD > 0, replace FD of status file of Frida thread by FD of legitimate app thread
- `openat()` syscall with patched path

Anti Frida

Frida threads detection

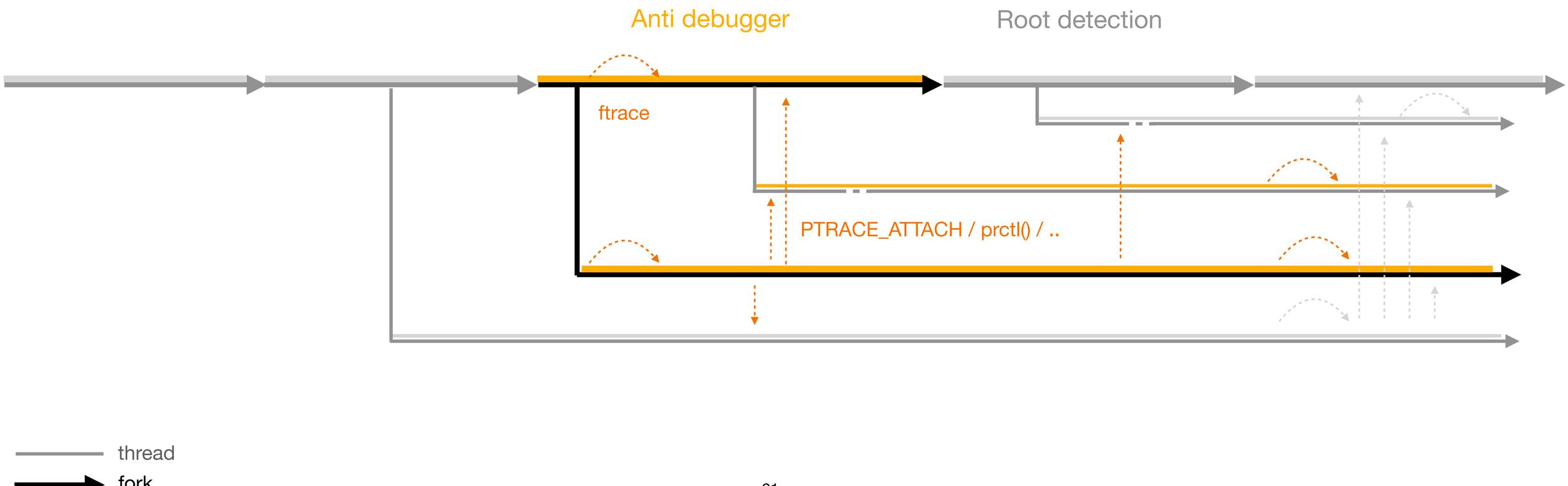
```
Interruptor.newAgentTracer({
    svc: {
        read: {
            onEnter: function(ctx){
                this.path = this.dxcFD[ctx.x0];
            },
            onLeave: function(ctx){
                if(this.path != null && this.path.match(/^\/proc\/[0-9]+\/task\/[0-9]+\/status$/g)>0){
                    this.path = null;

                    let res = Memory.scanSync(ctx.x1, ctx.x2.toInt32(), Interruptor.utils().toScanPattern('gmain'));
                    if(res.length > 0){
                        res.map( m => m.address.writeByteArray(Interruptor.utils().toByteArray("bindr")));
                        console.log("remove ‘gmain' pattern from resulting buffer");
                        return;
                    }

                    res = Memory.scanSync(ctx.x1, ctx.x2.toInt32(), Interruptor.utils().toScanPattern('gum-js'));
                    if(res.length > 0){
                        res.map( m => m.address.writeByteArray(Interruptor.utils().toByteArray("binder")));
                        console.log("remove ‘gum-js' pattern from resulting buffer");
                        return;
                    }
                }

            }
        }
    }
}).startOnLoad(/<YOUR_LIB>/g);
```

And now, a bunch of checks requiring …

CONSISTENCY

Between hooks and spoofed values

[anti-] Debugger


Anti debugger
Root detection
ftrace
PTRACE_ATTACH / prctl() / ..
thread
fork

Anti Debugger

Native « ftrace » detection

```
fd = openAt(procStatusStr);
if (fd == 0) {
  res = 0xffffffff;
}
else {
  o = (char *)0x0;
  local_f0 = 10;
  while( true ) {
    iVar3 = svc_read(fd,acStack496 + (long)o,0x100 - (long)o);
    if (0 < iVar3) {
      o = o + iVar3;
    }
    if (o == (char *)0x0) {
      res = 0;
      goto LAB_00109f50;
    }
    local_200[0] = acStack496;
    iVar3 = FUN_00109234(local_200,10,10);
    if (iVar3 == 0xd0359aa) break;
    cVar1 = local_200[0][-1];
    while (cVar1 != '\n') {
      cVar1 = *local_200[0];
      local_200[0] = local_200[0] + 1;
    }
    o = acStack496 + ((long)o - (long)local_200[0]);
    if (o != (char *)0x0) {
      memmove(acStack496,local_200[0],o);
    }
  }
  cVar1 = *local_200[0];
```

1. Decrypt string and search ‘TracePid’ row into ‘/proc/…/status‘

Hash of firsts ‘maxLen’ chars of line

=>

0xd0359aa = hash(‘TracerPid:’)

```
uint hashBeginOf(byte **bufferPtr,byte stopAtChar,int maxLenRead)

{
  uint hash;
  long o;
  byte *buffer;
  uint ch;

  if (maxLenRead < 1) {
    return 0;
  }
  buffer = *bufferPtr;
  o = 0;
  hash = 0;
  do {
    *bufferPtr = buffer + o + 1;
    ch = (uint)buffer[o];
    if (ch == stopAtChar) {
      return hash;
    }
    o = o + 1;
    hash = ch ^ (hash >> 0x1b | hash << 5);
  } while ((int)o < maxLenRead);
  return hash;
}
```

2. Browse content by hashing its entry and flag if ‘TracerPid: <val>’

(Note : there is not call to libc, everything is done with explicit SVC)

Anti Debugger

Java « ftrace » detection

```
 > INVOKE !  android.os.Debug.isDebuggerConnected(null) > false
 > INVOKE !  android.os.Debug.waitingForDebugger(null) < ...
 > INVOKE !  android.os.Debug.waitingForDebugger(null) > false
File (1) : new : /proc/self/status
 [JSONObject] New : {"IS_APP_DOING_WORK":"true"}
 String.contains( name: gitsecure.dgpay, tracerpid ) = false :
 String.contains( umask:        0077, tracerpid ) = false :
 String.contains( state:        s (sleeping), tracerpid ) = false :
 String.contains( tgid: 1161, tracerpid ) = false :
 String.contains( ngid: 0, tracerpid ) = false :
 String.contains( pid:  1161, tracerpid ) = false :
 String.contains( ppid: 3798, tracerpid ) = false :
 String.contains( tracerpid:    0, tracerpid ) = true :
 String.contains( uid:  10141   10141   10141   10141, tracerpid ) = false :
 String.contains( gid:  10141   10141   10141   10141, tracerpid ) = false :
```

```
 String.contains( mems_allowed_list:    0, tracerpid ) = false :
 String.contains( voluntary_ctxt_switches:      17605, tracerpid ) = false :
 String.contains( nonvoluntary_ctxt_switches:   64, tracerpid ) = false :
File (1) : new : /sys/kernel/debug/tracing/current_tracer
libc > access > /sys/kernel/debug/tracing/current_tracer 0x0
File (1) : new : /proc/sys/kernel/ftrace_enabled
libc > access > /proc/sys/kernel/ftrace_enabled 0xffffffffffffffff
```

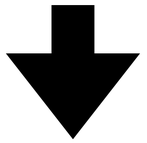
Anti Debugger

Prevent core dump from GDB / Debugger

```
start_RASP(){


  is_dumpable = prctl(PR_GET_DUMPABLE);
  if (is_dumpable != 1) {
    prctl(PR_SET_DUMPABLE,1);
  }
  …
```

TRUE if calling process can do core dump

Only a single process can dump data.
Allows the calling process to do core dump



GDB / Debugger cannot does core dump later

Anti Debugger

Bypass

```
prctl: {
    onEnter: function(ctx){
        this.cmd = ctx.x0.toUInt32();
        if(this.cmd === KAPI.CONST.PR_.OPT.PR_SET_DUMPABLE[0]){
            ctx.x0 = KAPI.CONST.PR_.OPT.PR_GET_DUMPABLE[0];
        }
    },
    onLeave: function(ctx){
        switch(this.cmd){
            case KAPI.CONST.PR_.OPT.PR_GET_DUMPABLE[0]:
            case KAPI.CONST.PR_.OPT.PR_SET_DUMPABLE[0]:
                ctx.x0 = 1;
                console.log("PRCTL SET/GET_DUMPABLE "+this.cmd+" returns "+ctx.x0);
                break;
            default:
                console.log("PRCTL * "+this.cmd+" returns "+ctx.x0);
                break;
        }
    }
},
```

Hook PRCTL system call with Interruptor and replace command

Anti Debugger

Start child process

Parent process

```
start_RASP(){

…

      pid = fork();
      if (pid != 0) {

        sigaction(0x11,(sigaction *)&handler,(sigaction *)0x0);
        JNI stuff here
        thread_args = 8;

      }
      else {
        if (pid == 0) {

          attach_n_restart_children();
          JNI stuff here with shared data
          start_RASP();
          gettime_hash((uint)java_global_ref ^ ftrace_detected);
          FUN_001028a4();
          return;
```

Anti Debugger

Start child process

Parent process

```
start_RASP(){

…

      pid = fork();
      if (pid != 0) {
        sigaction(0x11,(sigaction *)&handler,(sigaction *)0x0);
        JNI stuff here
        thread_args = 8;
      }
      else {
        if (pid == 0) {

          attach_n_restart_children();
          JNI stuff here with shared data
          start_RASP();
          gettime_hash((uint)java_global_ref ^ ftrace_detected);
          FUN_001028a4();
          return;
```

Anti Debugger

Prevent debugger to be attached to processes

Parent process

```
start_RASP(){

…

        pid = fork();
        if (pid != 0) {

          sigaction(0x11,(sigaction *)&handler,(sigaction *)0x0);
          JNI stuff here
          thread_args = 8;

        }
        else {
          if (pid == 0) {
```

```
attach_n_restart_children();
JNI stuff here with shared data
start_RASP();
gettime_hash((uint)java_global_ref ^ ftrace_detected);
FUN_001028a4();
return;
```

Executed by child process

Anti Debugger

Prevent debugger to be attached to processes

Child process

```
void attach_n_restart_children(void)

{
  __pid_t ppid;
  int iVar1;
  ulong success;

  ppid = getppid();
  success = ptrace_attach_all_thread(ppid);
  iVar1 = sem_post(_DAT_0011f248_sem);
  if ((success & 1) != 0) {
    start_restart_children(iVar1);
  }
                    /* WARNING: Subroutine does not return */
  _exit(0);
}
```

Anti Debugger

Prevent debugger to be attached to processes

Child process

```
ptrace_attach_all_thread(pid){

        . . .
        snprintf(parent_task,0x80,"/proc/%d/task",(ulong)(uint)pid);
        __dirp = opendir(parent_task);
        if (__dirp == (DIR *)0x0) { …. }

        dirEnts = readdir(__dirp);

        if (dirEnts != (dirent *)0x0) {
          do {
            if (dirEnts->d_name[0] != '.') {
              __pid = atoi(dirEnts->d_name);
              r = wrap_svc_ptrace(PTRACE_ATTACH, __pid, 0, 0);
              if (r == 0) goto on_attach_success;
            }
            dirEnts = readdir(__dirp);
            if (dirEnts == (dirent *)0x0) break;
          } while( true );
        }
        sig = closedir(__dirp);
```

Anti Debugger : attach each thread including main thread

Anti Debugger

Bypass

```
ptrace: {
    onEnter: function(ctx){
        this.cmd = ctx.x0.toUInt32();
        if(this.cmd === KAPI.CONST.PR_.OPT.PTRACE_ATTACH[0]){
            ctx.x0 = KAPI.CONST.PR_.OPT.PTRACE_DETACH[0];
        }
    },
},
```

Hook of PTRACE system call with Interruptor

PTRACE_ATTACH replaced by PTRACE_DETACH to avoid error

Try every time to avoid to create detectable side effects when you patch ptrace() args

3/
Automation and bypass

Bypass

Requirements

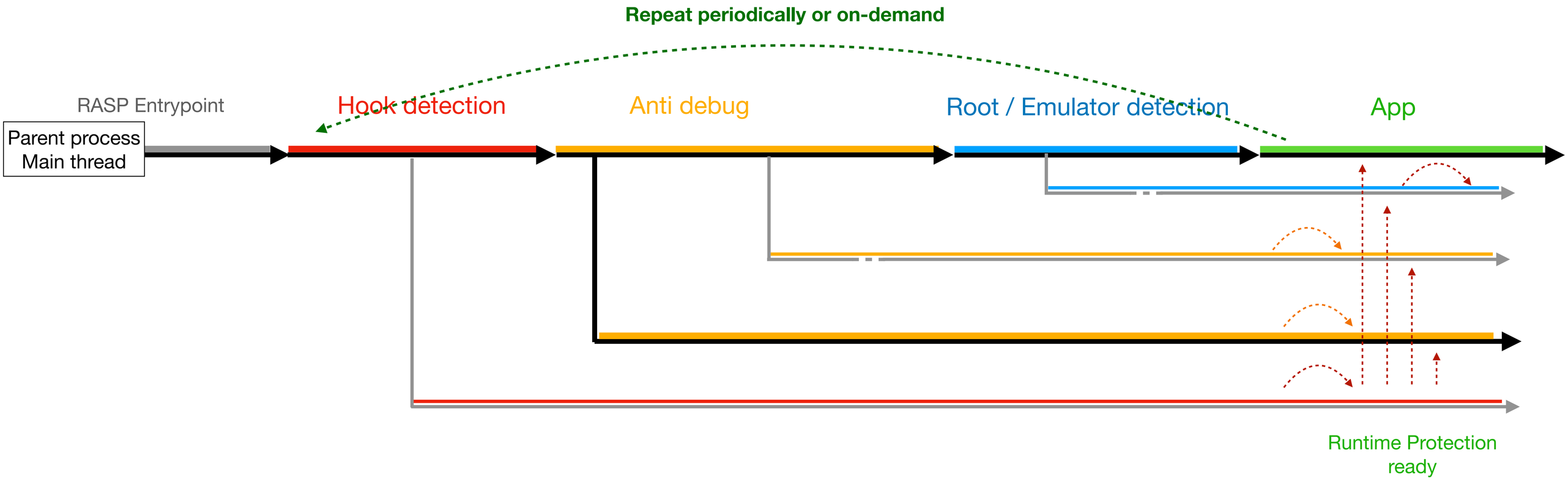
- Frida only ❤️
- Most of bypasses must be done from agent (repackage able)
- Universal (work for all apps)
- Autonomous « frida bot »
- Must work with official Frida binaries and Magisk

not only StrongR frida

or temporary root

The « Frida » Challenge

Remember …



The « Frida » Challenge

Challenge

Out of the box, without additional work, Frida cannot :

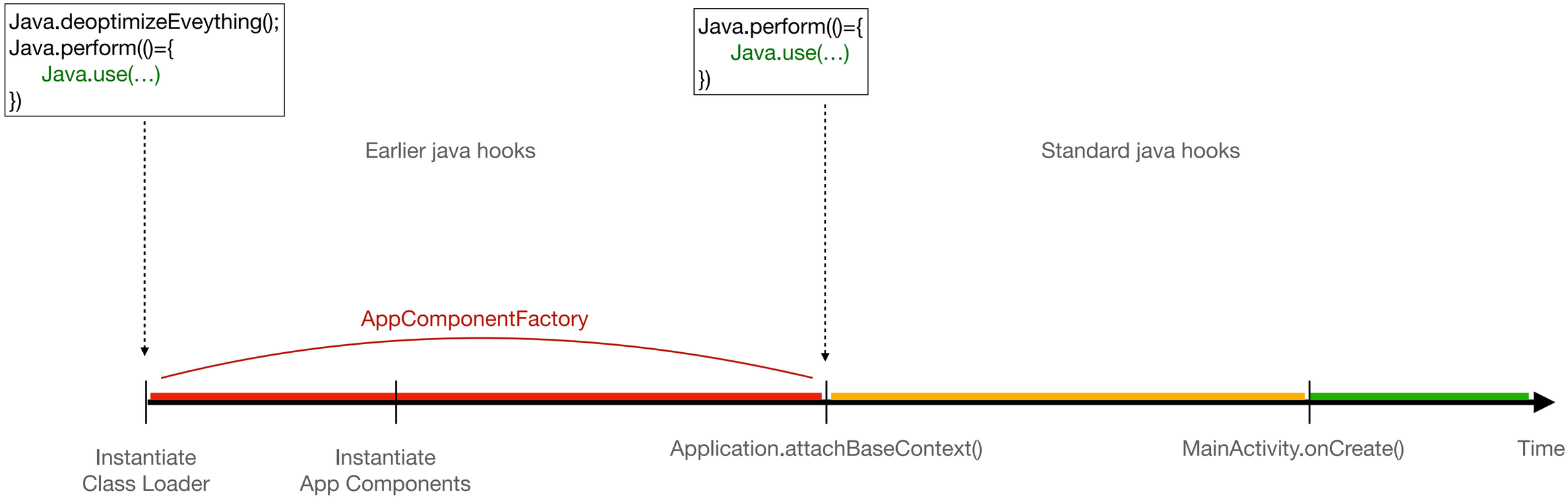
- Hook overridden AppComponentFactory’s methods
- Hook undefined Java method (methods loaded later)
- Hook native libs entrypoints (without patch)
- Hook instruction with follow thread
- Follow fork/clone (child gating) from app (instead of from host)
- Hook dynamically mapped ??X code prior to mapping

The « Frida » Challenge

- Act like a bot
- Stalk everything
- Fine grained hooking

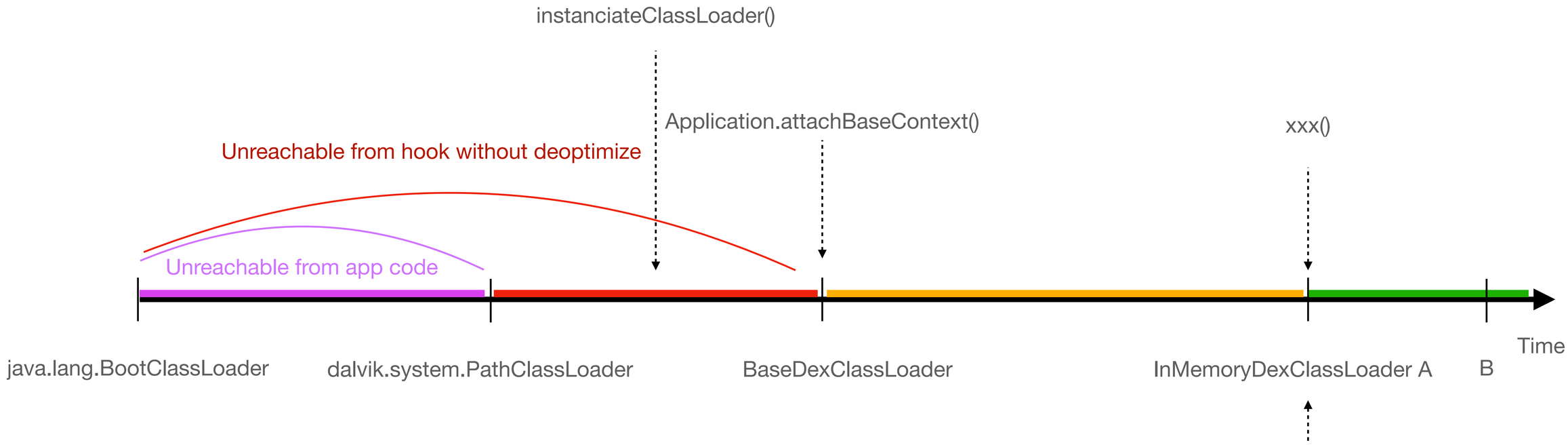
The « Frida » Challenge

Early java hooking



The « Frida » Challenge

Java hooking of dynamically loaded classes

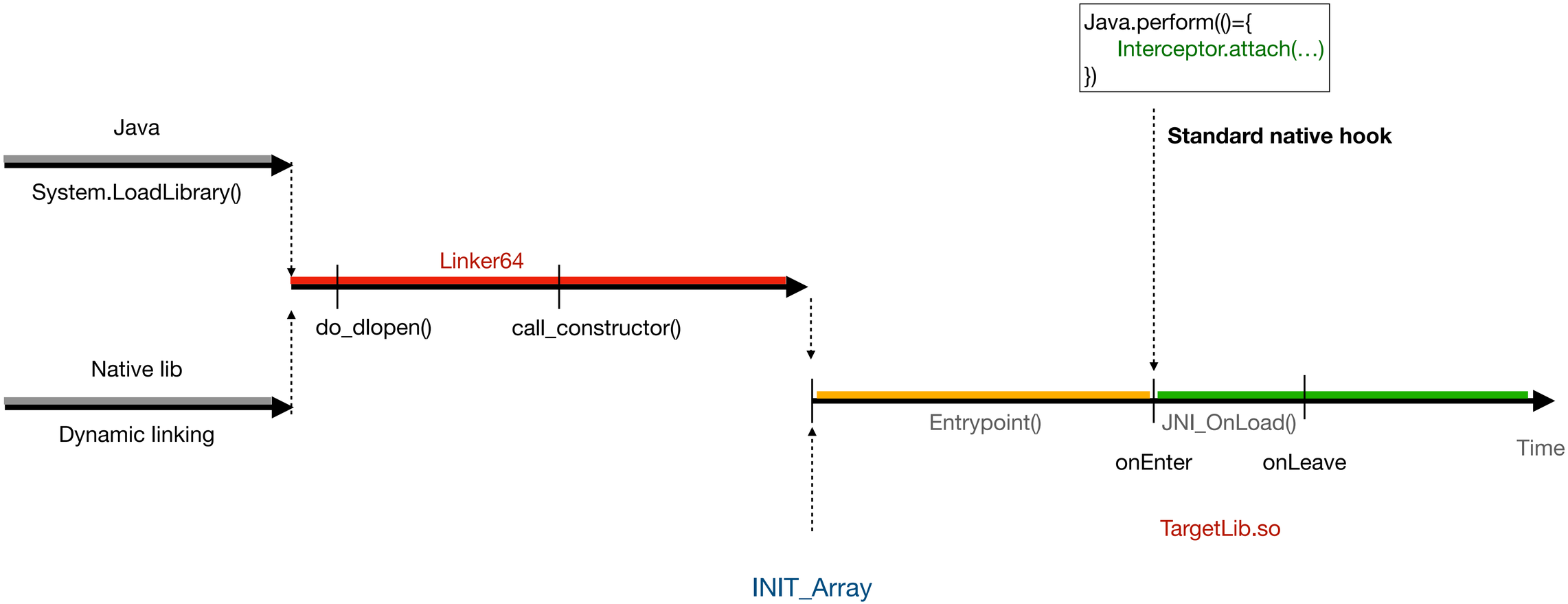



```
"<instance: java.lang.ClassLoader, $className: dalvik.system.PathClassLoader>"
"<instance: java.lang.ClassLoader, $className: java.lang.BootClassLoader>"
"<instance: java.lang.ClassLoader, $className: dalvik.system.PathClassLoader>"
"<instance: java.lang.ClassLoader, $className: dalvik.system.InMemoryDexClassLoader>"
"<instance: java.lang.ClassLoader, $className: dalvik.system.InMemoryDexClassLoader>"
```

Java.enumerateClassLoaderSync()

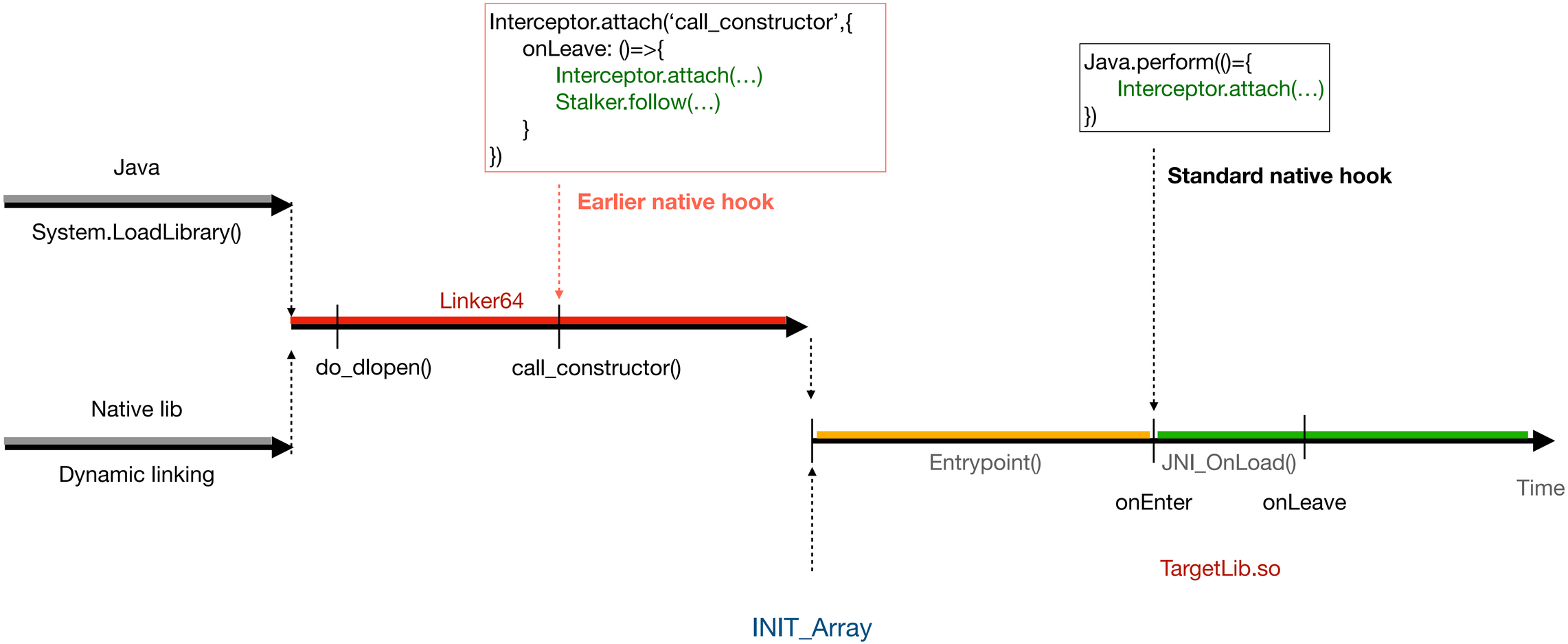
The « Frida » Challenge

Early native hooking



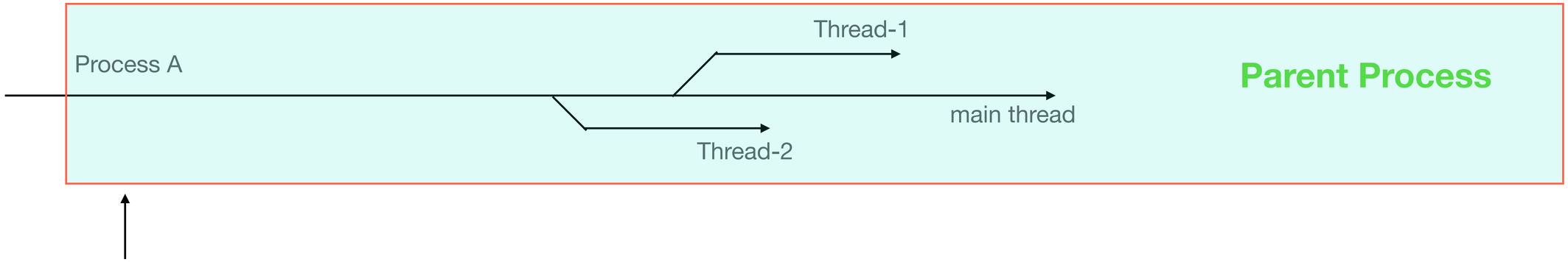
The « Frida » Challenge

Early native hooking



The « Frida » Challenge

Multi-thread

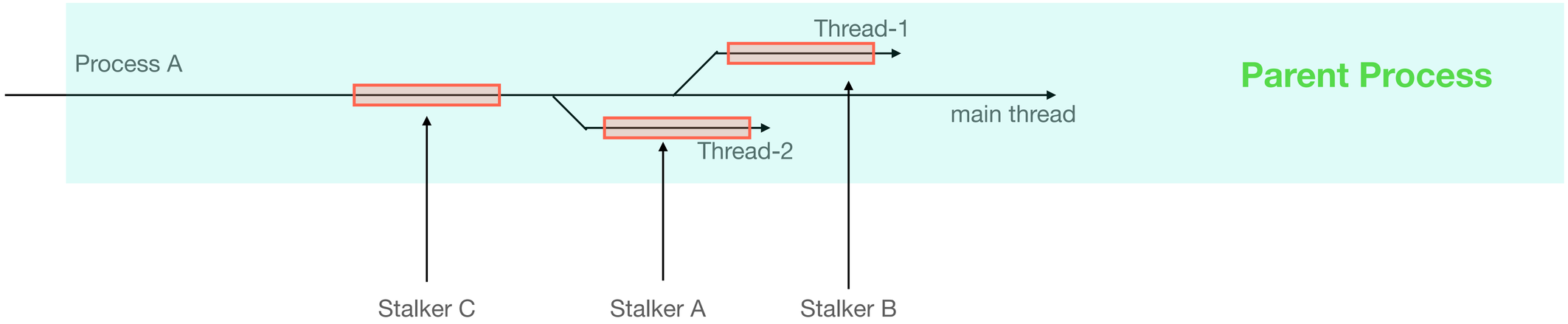


Java/Native hooks affect
shared executable memory:

Interceptor.attach(…)
Java.perform(…)

The « Frida » Challenge

Stalking multi-thread

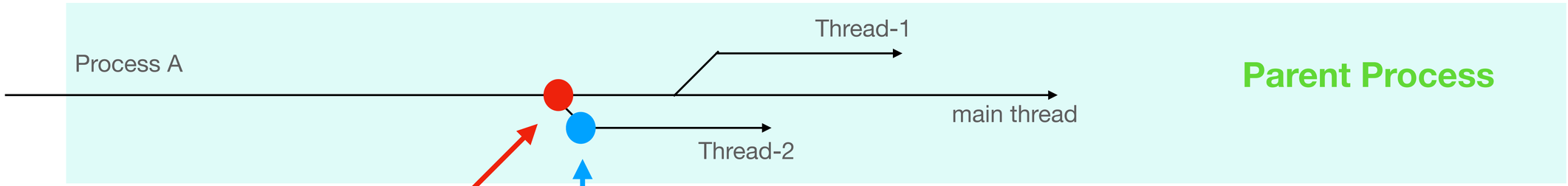


Don’t share instruction instrumentation done with Stalker

System call (SVC) hooks not shared

The « Frida » Challenge

Follow thread (over libc)



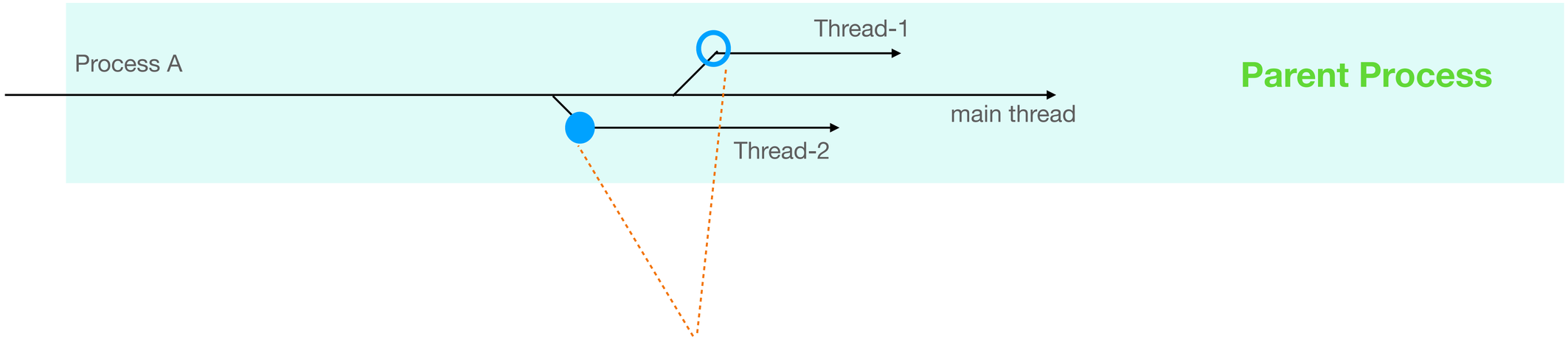

```
LinuxArm64InterruptorFactory.HOOKED_PTHREAD_ROUTINE = {};
Interceptor.attach( Module.findExportByName( moduleName: "libc.so", exportName: "pthread_create"), callbacksOrProbe: {
    onEnter: function(args : InvocationArguments ){
        let routine = args[2];
```

```
        if(routine != null && !LinuxArm64InterruptorFactory.HOOKED_PTHREAD_ROUTINE.hasOwnProperty(routine)){
            LinuxArm64InterruptorFactory.HOOKED_PTHREAD_ROUTINE[routine+""]=true;
            console.log("["+Process.findModuleByAddress(this.context.pc).name+"] Hooking routine : "+routine+" "
                +JSON.stringify(LinuxArm64InterruptorFactory.HOOKED_PTHREAD_ROUTINE));

            Interceptor.attach( routine, callbacksOrProbe: {
```

Thread start routine can be identical, Don’t hook it again !

The « Frida » Challenge

Follow thread (over libc)

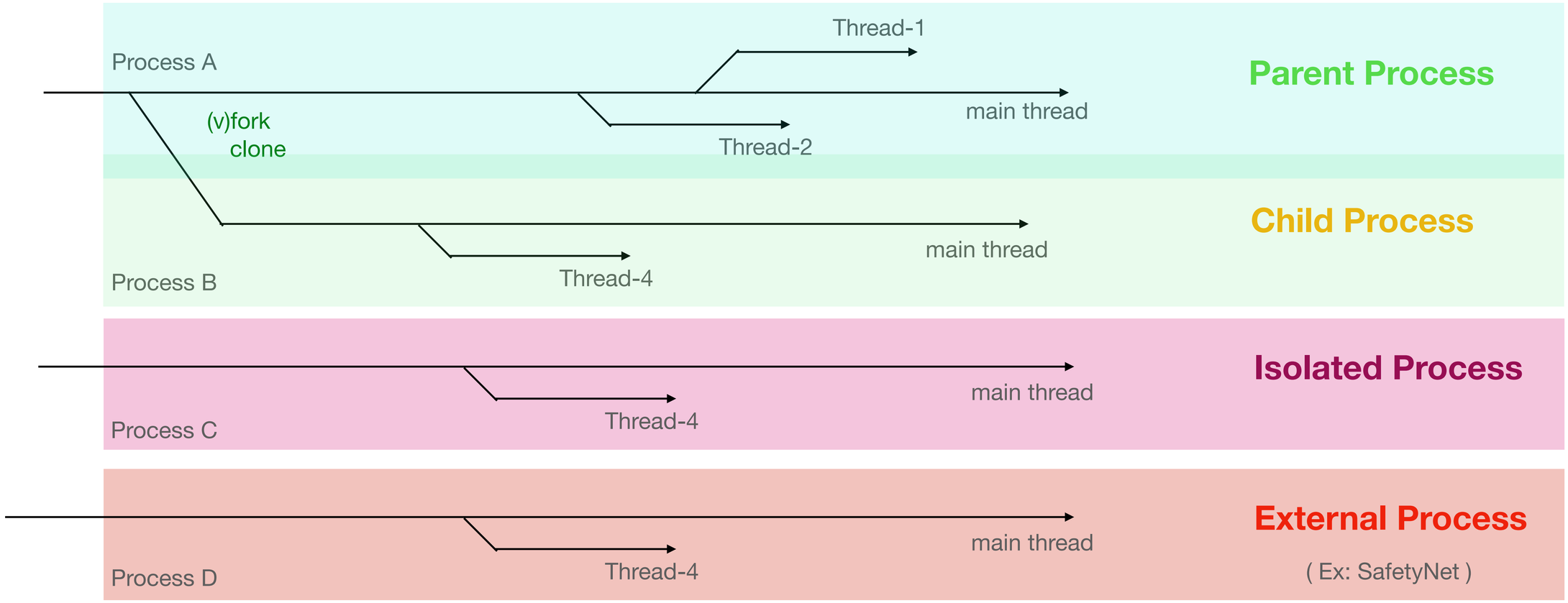


Thread start routine can be identical, Don’t hook it agin !

Thread start routine called from child

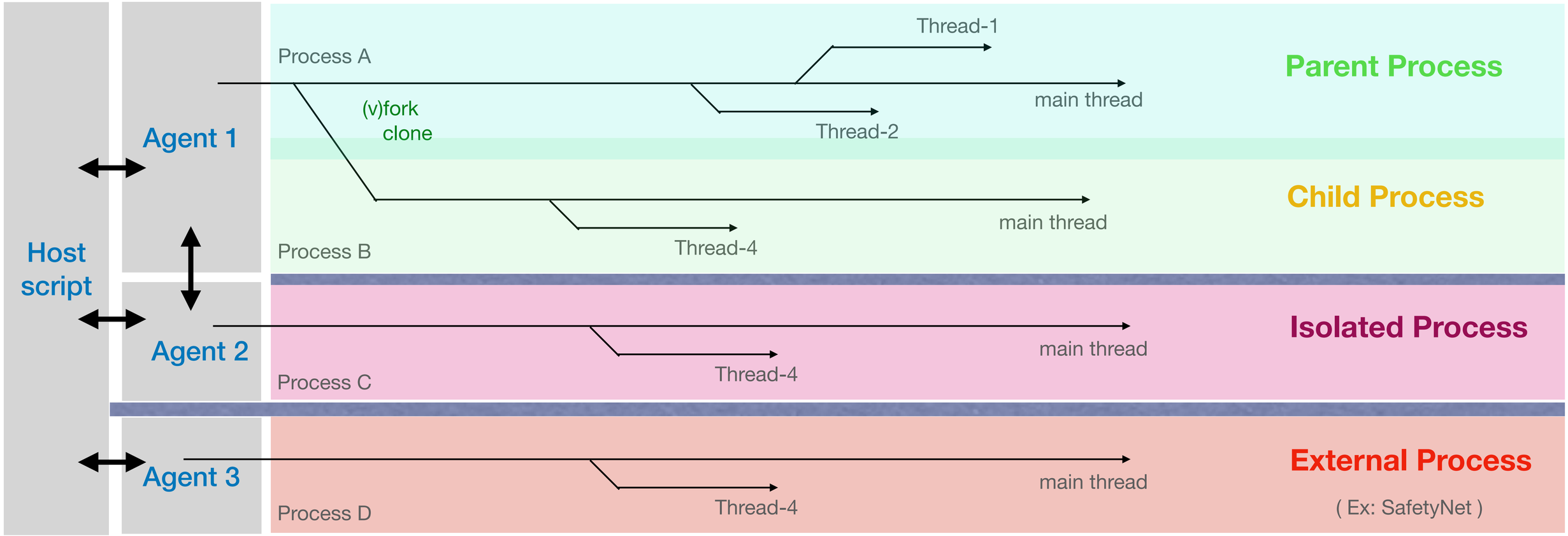
The « Frida » Challenge

Multi-process



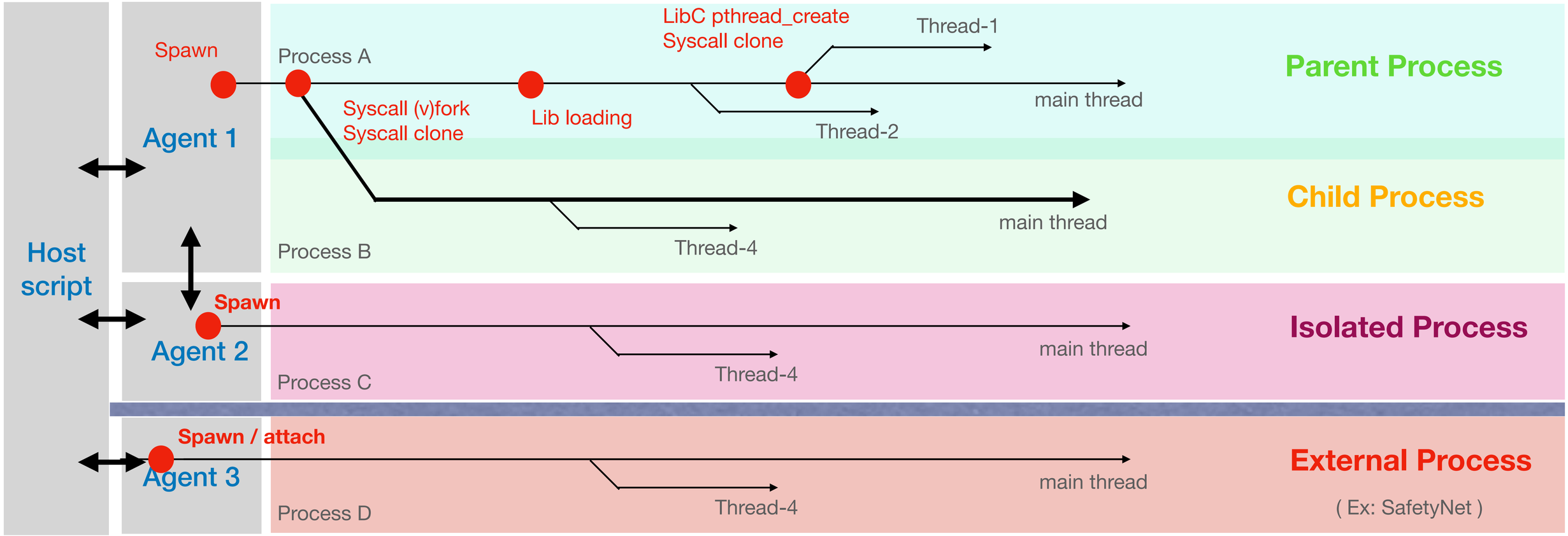
The « Frida » Challenge

Multi-process



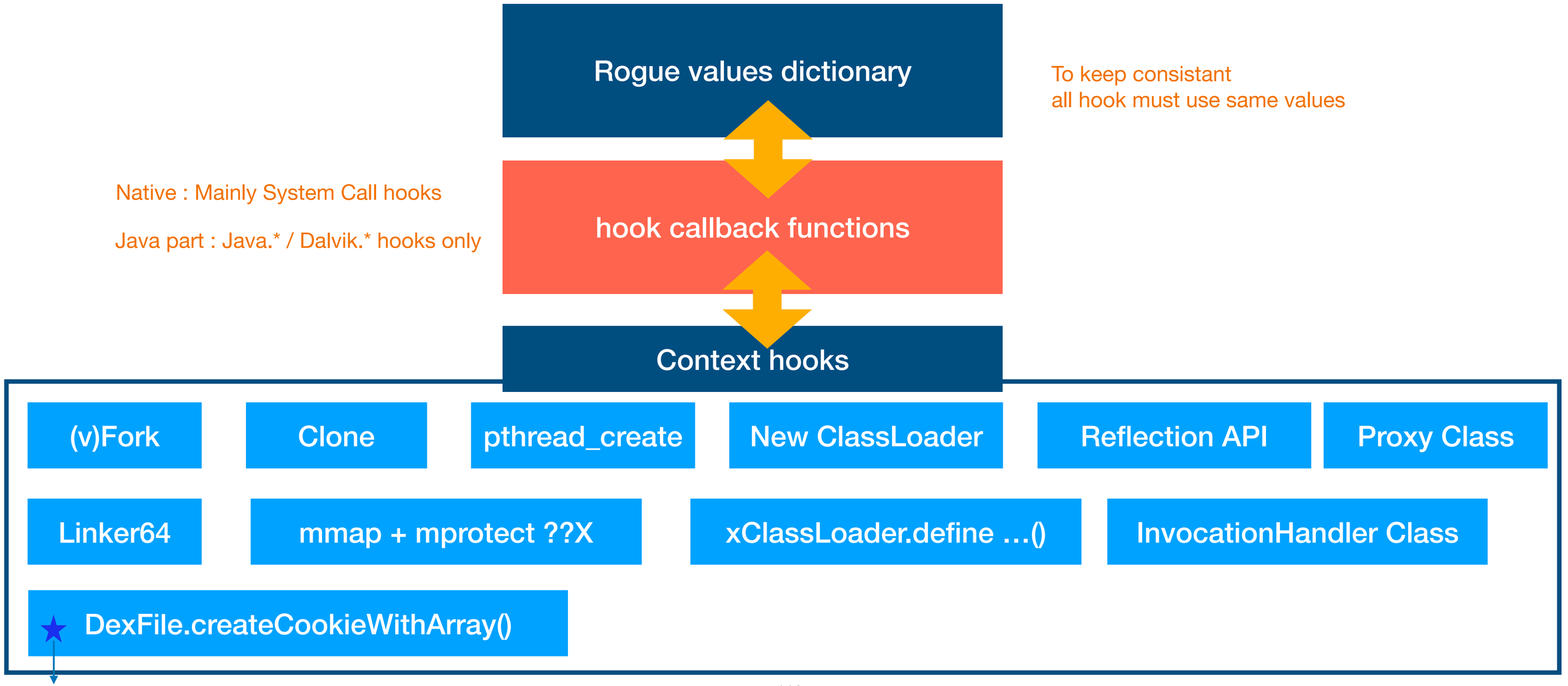
The « Frida » Challenge

Multi-process



The « Frida » Challenge

Final design



https://cs.android.com/android/platform/superproject/+/master:art/build/boot/hiddenapi/hiddenapi-max-target-o-low-priority.txt?q=createCookieWithArray&ss=android%2Fplatform%2Fsuperproject

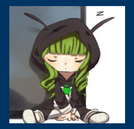
4/ Conclusion

Special thanks to

 @enovella_

 @Farenain

 @truffae

 @horsicq

Q&A