One Step Before Game Hackers

-- Instrumenting Android Emulators

Defcon 26
nevermoe

Self Introduction

- nevermoe (@n3v3rm03, i [at] nevermoe.com)
- Chinese in Japan
- Security engineer in DeNA Co., Ltd.
- Love playing / hacking games

Agenda

- Background
- Emulator Internal
- Hooking
- Demo
- Conclusion

Background:
Game Cheating Threat Model

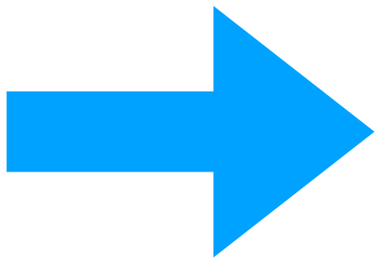
Full Control?	Users	Cheaters	Game Vendors
PC	YES	YES	YES
Mobile	(Normally) No	YES	No

Background:
Mobile Game Cheating Business Model

- Is there an easy way to distribute cheating tools?
  - Android emulators!
    - Unified environment
    - Already or easily rooted

Background:
Mobile Game Cheating Business Model

- Cheating on emulators
  - Popular: Touch simulation (e.g. Mobile Anjian)
  - Why are there no hooking tools?
    - Game codes are usually native
    - Commercial emulators use Intel Houdini for arm-x86 translation in native code



Hooking solution is not obvious

Background: Purpose

- Enable hooking on commercial Android emulators!

Emulator Internal: Targets

	Client Ver.	Android Ver.	Houdini Ver.
BlueStacks	3.56.73.1817	4.4.2	4.0.8.45720
NOX	6.0.5.2	4.4.2	4.0.8.45720
NOX	6.0.5.2	5.5.1	5.0.7b_x.48396
LeiDian	2.0.54	5.5.1	5.0.7b_x.48396
MEmu	5.3.1	5.5.1	5.0.7b_x.48396

Emulator Internal: Command Line Binary

```
// file: enable_nativebridge.sh

cd $binfmt_misc_dir
if [ -e register ]; then
  echo ':arm_exe:M::\\x7f\\x45\\x4c\\x46\\x01\\x01\\x01\\x00\\x00\\x00\\x00\
\x00\\x00\\x00\\x00\\x00\\x02\\x00\\x28::'"/system/lib/arm/houdini:P" > register
  echo ':arm_dyn:M::\\x7f\\x45\\x4c\\x46\\x01\\x01\\x01\\x00\\x00\\x00\\x00\
\x00\\x00\\x00\\x00\\x00\\x03\\x00\\x28::'"/system/lib/arm/houdini:P" > register
fi
```

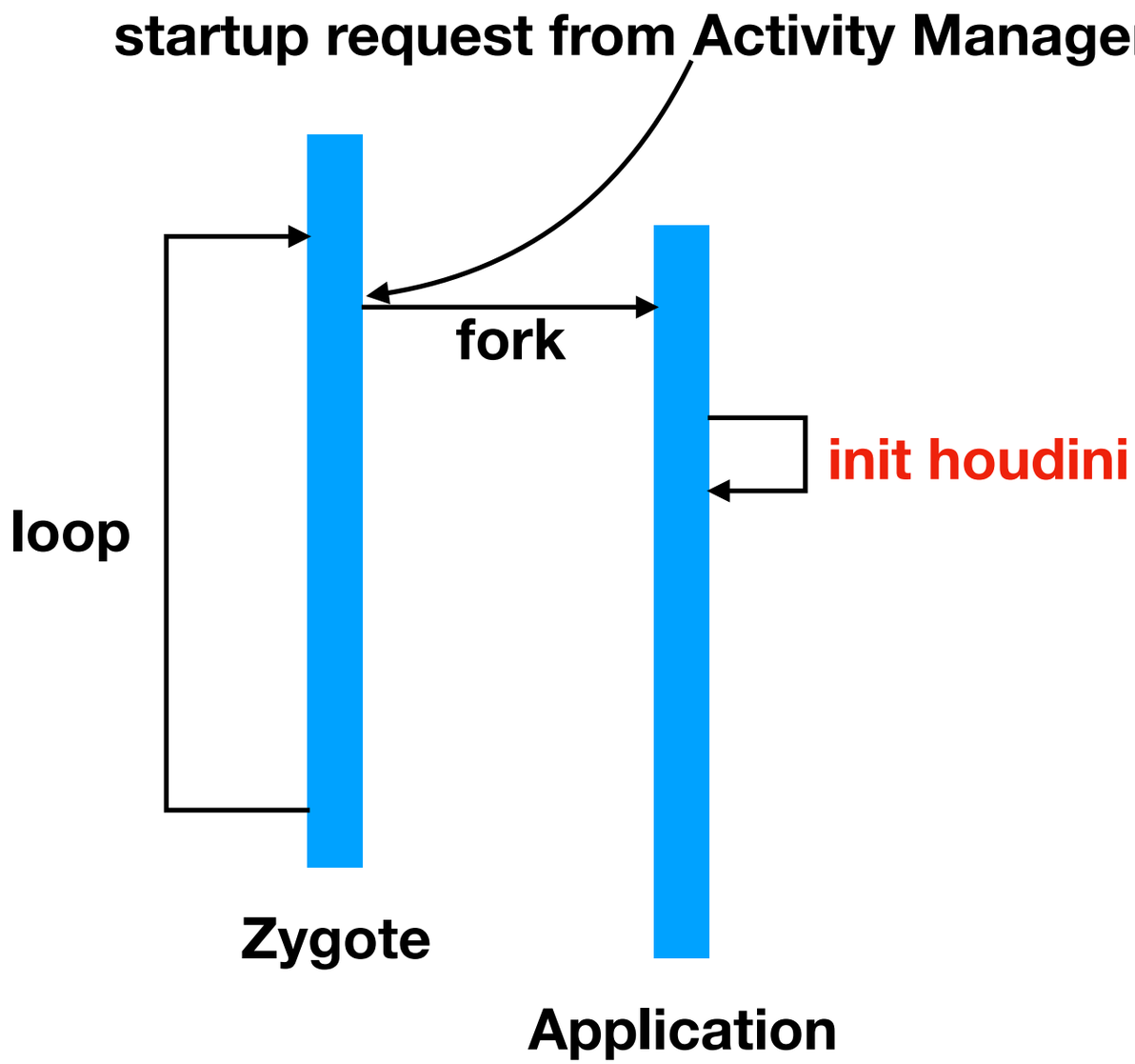
- **Hook it**
  - LD_PRELOAD=libinject_arm.so ./target_exe_arm ✅
  - ptrace(x86) target_pid ❌
  - ptrace(arm) target_pid ❌

Emulator Internal: Java Application

- Is LD_PRELOAD useful in Java application hooking?

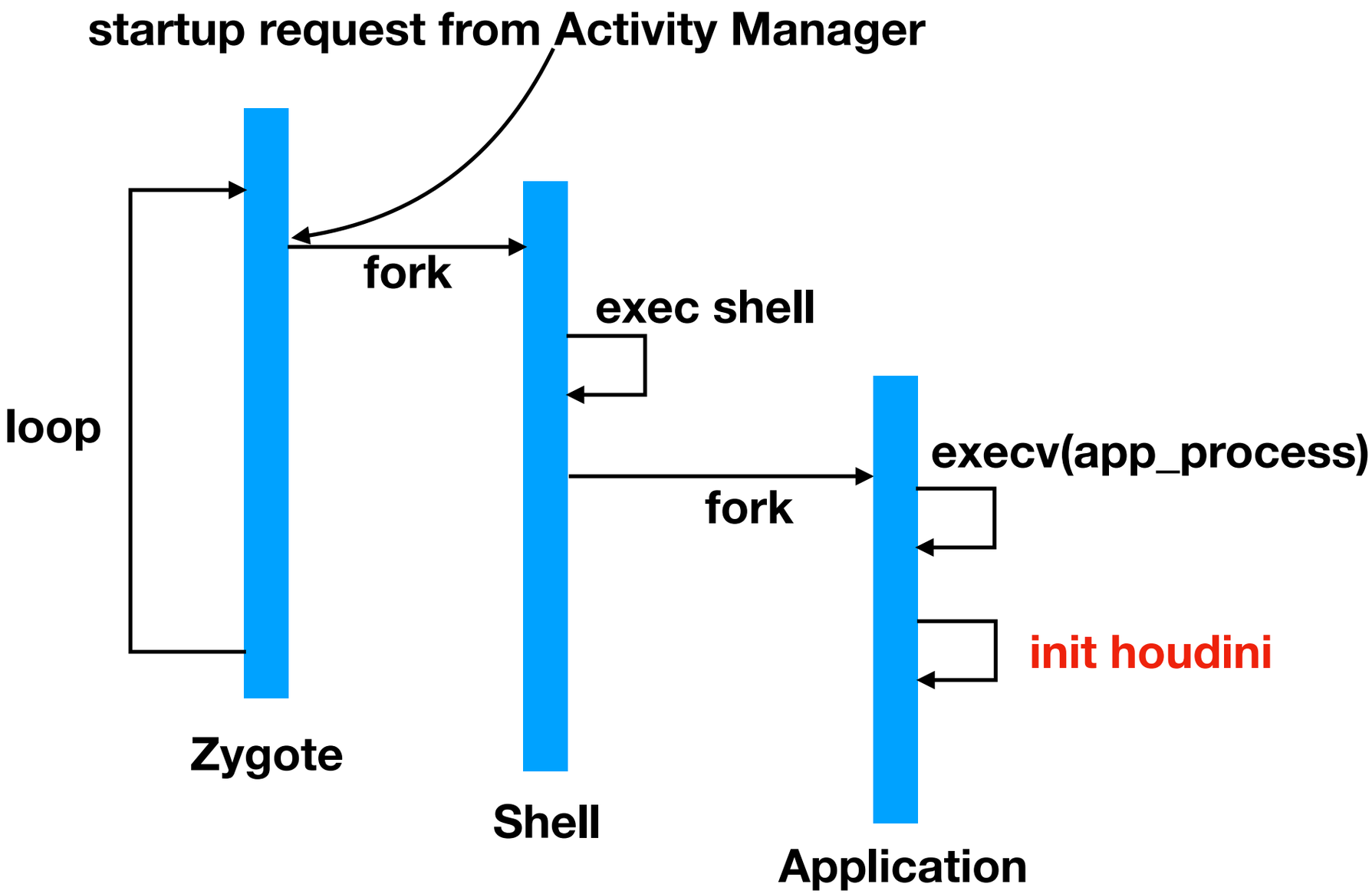
Emulator Internal: Java Application

- Normal startup



Emulator Internal: Java Application

- Start with “wrap” system property
  - setprop wrap.com.nevermoe.example LD_PRELOAD=libinject.so



Emulator Internal: Java Application

- Start with “wrap” system property

runSelectLoop() — frameworks/base/core/java/com/android/internal/os/ZygoteInit.java
- runOnce() — frameworks/base/core/java/com/android/internal/os/ZygoteConnection.java
  - forkAndSpecialize() — frameworks/base/core/java/com/android/internal/os/Zygote.java
    - ForkAndSpecializeCommon() — frameworks/base/core/jni/com_android_internal_os_Zygote.cpp
    - handleChildProc() — frameworks/base/core/java/com/android/internal/os/ZygoteConnection.java
      - execApplication() — frameworks/base/core/java/com/android/internal/os/WrapperInit.java

```
public static void execApplication(String invokeWith, String niceName,
        int targetSdkVersion, FileDescriptor pipeFd, String[] args) {
    StringBuilder command = new StringBuilder(invokeWith);
    command.append(" /system/bin/app_process /system/bin --application");
    if (niceName != null) {
        command.append(" '--nice-name=").append(niceName).append("'");
    }
    command.append(" com.android.internal.os.WrapperInit ");
    command.append(pipeFd != null ? pipeFd.getInt$() : 0);
    command.append(' ');
    command.append(targetSdkVersion);
    Zygote.appendQuotedShellArgs(command, args);
    Zygote.execShell(command.toString());
}
```

Emulator Internal: Java Application

- Start with "wrap" property

x86 ↓ /system/bin/sh -c LD_PRELOAD=libinject_arm.so ↓ arm \
/system/bin/app_process /system/bin --application \
'--nice-name=com.nevermoe.myapp' \
com.android.internal.os.WrapperInit 48 21 \
'android.app.ActivityThread'

- Won't do the trick

Emulator Internal: Init Houdini

- (Android 5.1.1 / 4.4.2) app_process -- Start as Zygote

main() — frameworks/base/cmds/app_process/app_main.cpp
- AndroidRuntime::start() — frameworks/base/core/jni/AndroidRuntime.cpp
  - AndroidRuntime::startVm() — frameworks/base/core/jni/AndroidRuntime.cpp
    - JNI_CreateJavaVM() — art/runtime/jni_internal.cc
      - Runtime::Start() — art/runtime/runtime.cc
  - ZygoteInit::main() — frameworks/base/core/java/com/android/internal/os/ZygoteInit.java

Emulator Internal: Init Houdini

- (Android 5.1.1) Zygote fork process

runSelectLoop() — frameworks/base/core/java/com/android/internal/os/ZygoteInit.java
- runOnce() — frameworks/base/core/java/com/android/internal/os/ZygoteConnection.java
  - forkAndSpecialize() — frameworks/base/core/java/com/android/internal/os/Zygote.java
    - ForkAndSpecializeCommon() — frameworks/base/core/jni/com_android_internal_os_Zygote.cpp
      - callPostForkChildHooks() — frameworks/base/core/java/com/android/internal/os/Zygote.java
        - postForkChild() — libcore/dalvik/src/main/java/dalvik/system/ZygoteHooks.java
          - ZygoteHooks_nativePostForkChild() —art/runtime/native/dalvik_system_ZygoteHooks.cc
            - Runtime::DidForkFromZygote — art/runtime/runtime.cc
              - InitializeNativeBridge — art/runtime/native_bridge_art_interface.cc
                - InitializeNativeBridge — system/core/libnativebridge/native_bridge.cc
    - handleChildProc() — frameworks/base/core/java/com/android/internal/os/ZygoteConnection.java
      - zygoteInit() — frameworks/base/core/java/com/android/internal/os/RuntimeInit.java

Emulator Internal: Init Houdini

- Android 5.1.1

```
// Native bridge interfaces to runtime.
struct NativeBridgeCallbacks {
    uint32_t version;
    bool (*initialize)(const NativeBridgeRuntimeCallbacks* runtime_cbs, const char* private_dir,
    void* (*loadLibrary)(const char* libpath, int flag);
    void* (*getTrampoline)(void* handle, const char* name, const char* shorty, uint32_t len);
    bool (*isSupported)(const char* libpath);
    const struct NativeBridgeRuntimeValues* (*getAppEnv)(const char* instruction_set);
    bool (*isCompatibleWith)(uint32_t bridge_version);
    NativeBridgeSignalHandlerFn (*getSignalHandler)(int signal);
};
```

```
// libhoudini.so
.data:00379198  NativeBridgeItf  dd 2
.data:0037919C                   dd offset sub_1BD070
.data:003791A0                   dd offset sub_1BCC80
.data:003791A4                   dd offset sub_1BCD60
.data:003791A8                   dd offset sub_1BCEC0
.data:003791AC                   dd offset sub_1BCF40
.data:003791B0                   dd offset sub_1BCF90
.data:003791B4                   dd offset sub_1BCFE0
```

Emulator Internal: Init Houdini

- Android 4.4.2

```
// file: platform/dalvik/vm/Native.cpp
dvmLoadNativeCode()
  ├ houdini::hookDlopen()
  │   └ houdiniHookInit()
  └ houdini::hookJniOnload()
```

```
hookDlopen()
{
    v3 = dlopen((const char *)this, (int)a2);
    if ( v3 )
        return v3;

    else
        houdiniHookInit();
}
```

```
houdiniHookInit()
{
    v15 = dword_4F2F84;
    *(_DWORD *)(v15 + 8) = dlsym(handle, "dvm2hdDlopen");
    v16 = dword_4F2F84;
    *(_DWORD *)(v16 + 12) = dlsym(handle, "dvm2hdDlsym");
    v17 = dword_4F2F84;
    *(_DWORD *)(v17 + 20) = dlsym(handle, "dvm2hdNeeded");
    v18 = dword_4F2F84;
    *(_DWORD *)(v18 + 16) = dlsym(handle, "dvm2hdNativeMethodHelper");
    v19 = dword_4F2F84;
    *(_DWORD *)(v19 + 24) = dlsym(handle, "androidrt2hdCreateActivity");
}
```

Emulator Internal: Houdini License

- Genymotion
  - No houdini provided
- Bluestacks
  - lib3btrans.so == libhoudini.so

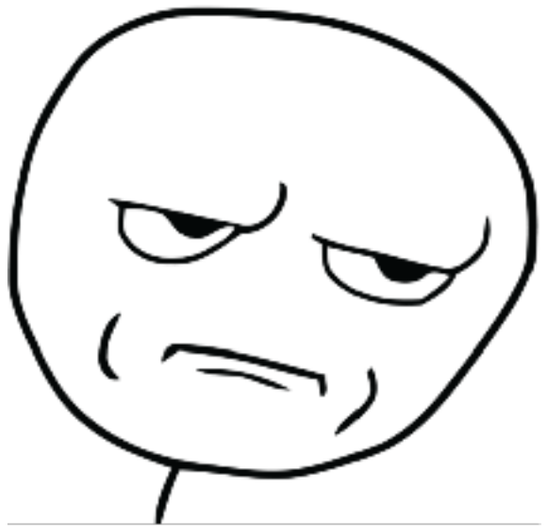
```
v12 = dlopen("/system/lib/lib3btrans.so", 0);
handle = v12;
v3 = v22;
if ( v12 )
{
  v13 = dword_4F2F84;
  *(_DWORD *)dword_4F2F84 = v12;
  *(_DWORD *)(v13 + 4) = dlsym(v12, "dvm2hdInit");
  v14 = *(int (__cdecl **)(void **))(dword_4F2F84 + 4);
  if ( !v14 )
  {
    dlerror();
    v21 = "Cannot find symbol dvm2hdInit, please check the libhoudini library is correct: %s!¥n";
```

- NOX
  - packed libdvm.so

Emulator Internal: Houdini License

- Genymotion
  - No houdini provided
- Bluestacks
  - lib3btrans.so == libhoudini.so

```
v12 = dlopen("/system/lib/lib3btrans.so", 0);
handle = v12;
v3 = v22;
if ( v12 )
{
  v13 = dword_4F2F84;
  *(_DWORD *)dword_4F2F84 = v12;
  *(_DWORD *)(v13 + 4) = dlsym(v12, "dvm2hdInit");
  v14 = *(int (__cdecl **)(void **))(dword_4F2F84 + 4);
  if ( !v14 )
  {
    dlerror();
    v21 = "Cannot find symbol dvm2hdInit, please check the libhoudini library is correct: %s!¥n";
```

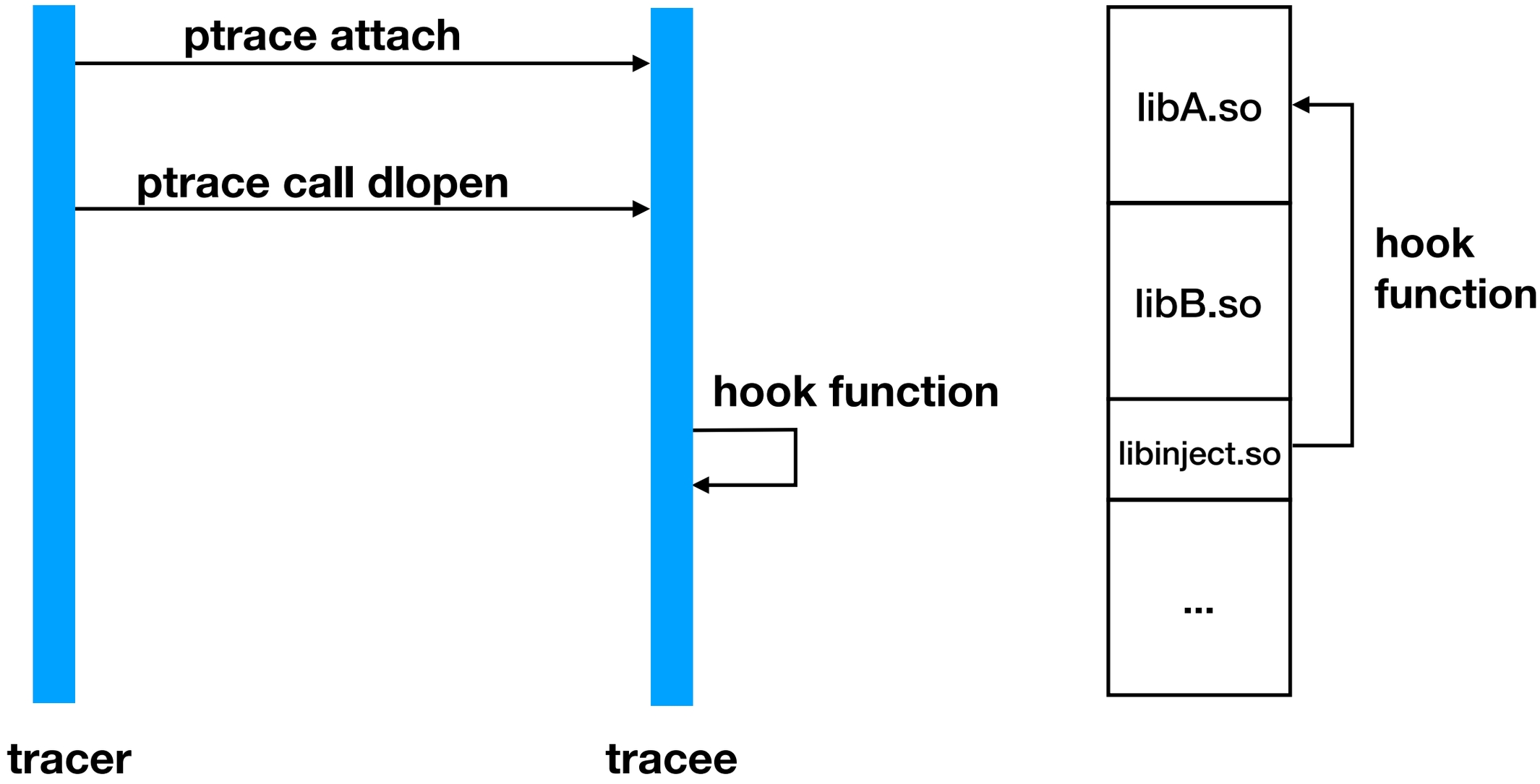


- NOX
  - packed libdvm.so

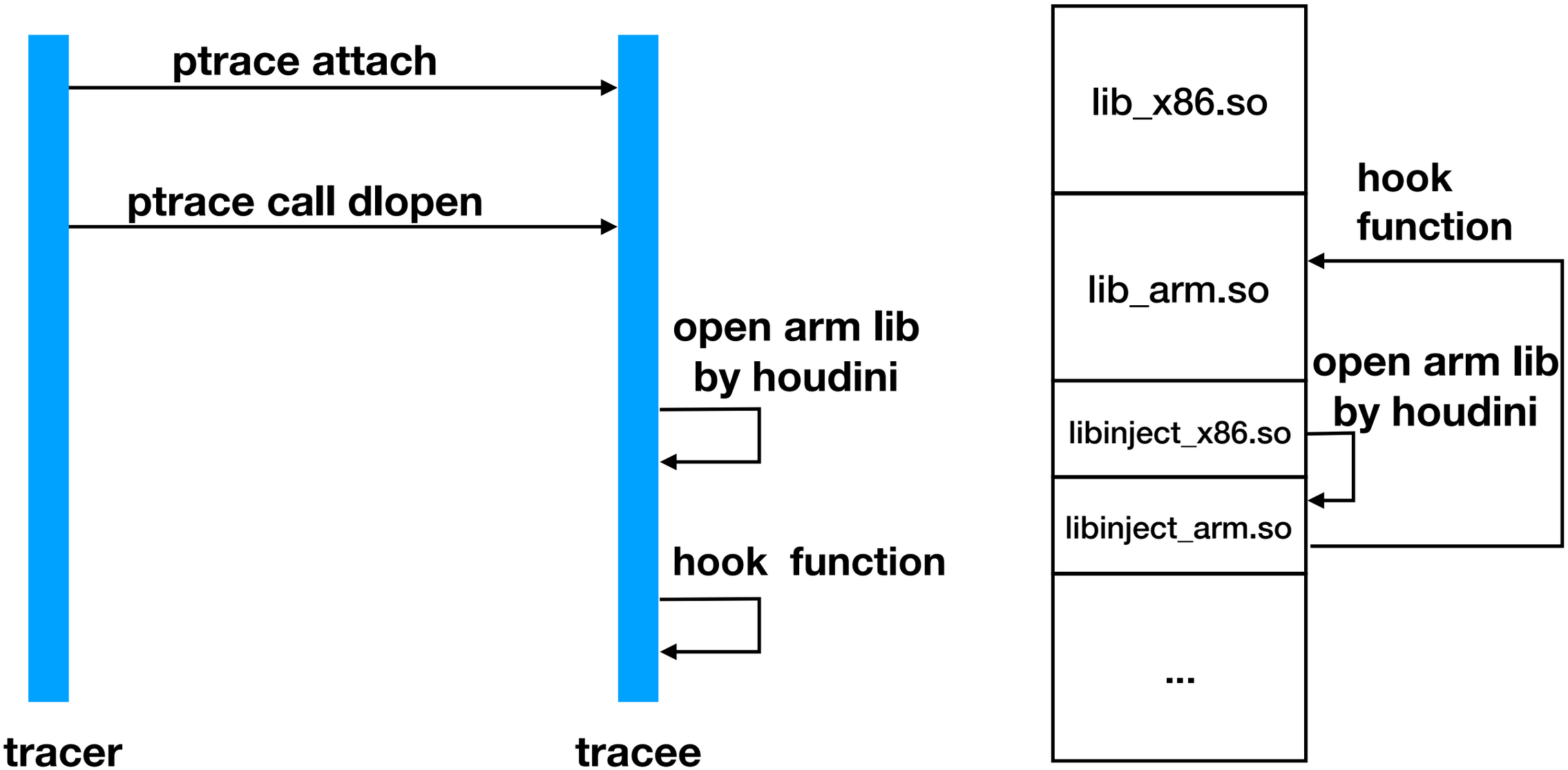
Hooking:
Existing Hooking Framework

- Xposed
  - Only Java Layer (Discuss this later)
  - Substitute app_process to load its own jar file
- Frida
  - Omnipotent
  - “I'm afraid NOX is unsupported. Please use a stock emulator or real device, or help us fix this. It's not a priority for me personally so unless somebody helps out, NOX support will not happen. :-/”
- Substrate (on Android)
  - Fake liblog.so
  - Outdated

Hooking:
Normal Approach



Hooking on Emulator: (A) Utilize Houdini



Hooking on Emulator: (B) Utilize Xposed

```
public class NativeHook {
   static{
      System.load("/path/to/libinject_arm.so");
   }
   public native static void initNativeHook();
}
```

```
   findAndHookMethod("android.app.Application", lpparam.classLoader,
"onCreate", new XC_MethodHook() {
          @Override
          protected void beforeHookedMethod(MethodHookParam param) throws
Throwable {
             NativeHook.initNativeHook();
          }
          @Override
          protected void afterHookedMethod(MethodHookParam param) throws
Throwable {
          }
   });
```

Demo

- Method A: github.com/nevermoe/EHook
  - stable with ptrace
- Method B: github.com/nevermoe/XEHook
  - Early trace
  - Does not trigger anti-debug mechanism

Usage:

```
void real_init_func()
{
    hook_by_addr(&h1, "nb/libc.so", target_addr, hook_target);
    hook_by_name(&h2, "nb/libc.so", "recvfrom", hook_recvfrom);
}
```

Conclusion

- Mobile game is getting more popular as well as cheating
- Cheating patterns change as the technique develops
- To cooperate with emulator vendors, or not to, that is the question
  - Advertising on emulator and targeting the emulator users?
  - Restricting emulator users?
  - Putting emulators users to a dedicated server?
- Let's see what's going to change

Thank You!