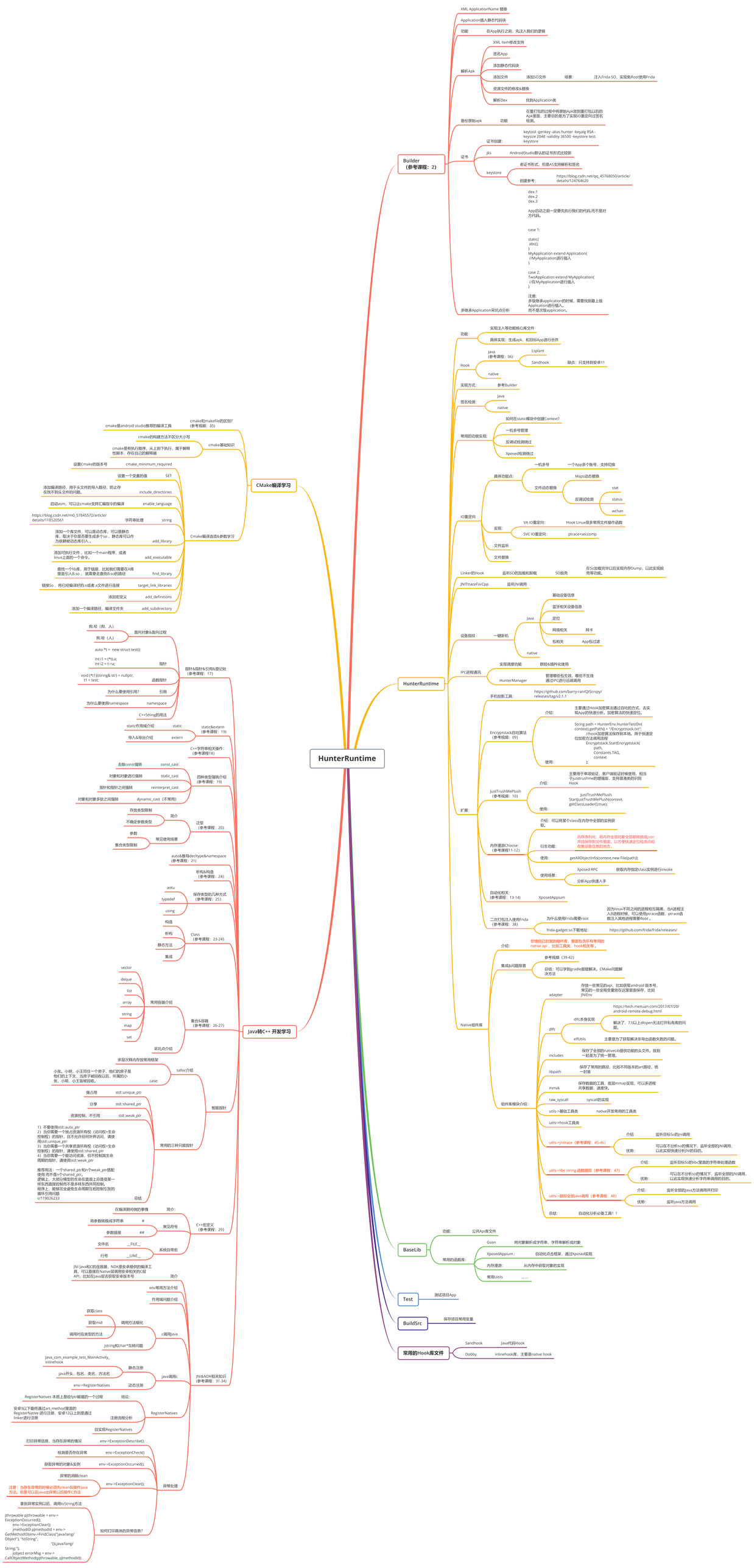
HunterRuntime
Builder
（参考课程：2）
XML ApplicationName 替换
Application插入静态代码块
功能
在App执行之前，先注入我们的逻辑
解析Apk
XML Item修改支持
签名App
添加静态代码块
添加文件
添加SO文件
场景：
注入Frida SO，实现免Root使用Frida
资源文件的修改&替换
解析Dex
找到Application类
备份原始apk
功能
在重打包的过程中将原始Apk放到重打包以后的Apk解压，主要目的是为了实现C重定向过签名检测。
证书
证书创建：
keytool -genkey -alias hunter -keyalg RSA -keysize 2048 -validity 36500 -keystore test.keystore
jks
AndroidStudio默认的证书形式比较新
keystore
老证书形式，但是AS支持解析和签名
创建参考：
https://blog.csdn.net/qq_45768050/article/details/124764620
多继承Application采坑点分析
dex.1
dex.2
dex.3
App启动之前一定要先执行我们加的代码,而不是对方代码。
case 1:
static{
abc();
}
MyApplication extend Application{
//MyApplication进行插入
}
case 2:
TwoApplication extend MyApplication{
//在MyApplication进行插入
}
注意:
多级继承application的时候，需要找到最上级Application进行插入，
而不是次级application。
HunterRuntime
功能：
实现注入等功能核心库文件
具体实现：生成apk，和目标App进行合并
Hook
Java
（参考课程：06）
Lsplant
Sandhook
缺点：只支持到安卓11
native
实现方式：
参考Builder
签名检测：
java
native
常用的功能实现：
如何在static模块中创建Context?
一机多号管理
反调试检测触发
Xposed检测触发
IO重定向
具体功能点：
一机多号
一个App多个账号，支持切换
文件动态替换
Maps动态替换
反调试检测
stat
status
wchan
实现：
VA IO重定向：
Hook Linux很多常用文件操作函数
SVC IO重定向：
ptrace+seccomp
文件监听
文件替换
Linker的Hook
监听SO的加载和卸载
SO脱壳
在So加载完毕以后实现内存Dump，以此实现脱壳等功能。
JNITraceForCpp
监听JNI调用
设备指纹
一键新机
java
基础设备信息
蓝牙相关设备信息
定位
网络相关
网卡
包相关
App包过滤
native
IPC进程通讯
实现调度功能
HunterManager
群控&插件化使用
管理哪些包生效，哪些不生效
通过IPC进行远端调用
扩展：
手机投影工具：
https://github.com/barry-ran/QtScrcpy/releases/tag/v2.1.1
Encryptstack自吐算法
（参考视频：09）
介绍：
主要通过Hook加密算法通过自吐的方式，去实现App的快速分析，加密算法的快速定位。
String path = HunterEnv.HunterTestDir(context).getPath() + "/Encryptstack.txt";
//hook加密算法保存到本地。用于快速定位加密方法调用流程
Encryptstack.StartEncryptstack(
path,
Constants.TAG,
context
);
使用：
JustTrushMePlush
（参考视频：10）
介绍：
主要用于单项验证，客户端验证时候使用，相当于justtrushme的增强版，支持黑毒类的识别Hook
JustTrushMePlush.
StartJustTrushMePlush(context.
getClassLoader(),true);
使用：
内存遍游Choose：
（参考课程11-12）
介绍：可以将某个class在内存中全部的实例获取。
衍生功能：
内存序列化：将内存全部对象全部都转换成json并且保存到文件里面，以方便快速定位检测点和收集设备信息的地方。
使用：
getAllObjectInfo(context,new File(path));
使用场景：
Xposed RPC
获取内存指定class实例进行invoke
分析App快速入手
自动化相关：
（参考课程：13-14）
XposedAppium
二次打包注入使用Frida
（参考课程：38）
为什么使用Frida需要root
因为linux不同之间的进程相互隔离，当A进程注入B进程时候，可以使用ptrace函数，ptrace函数注入其他进程需要root。
frida-gadget.so下载地址：
https://github.com/frida/frida/releases/
Native组件库
介绍：
珍惜自己封装的组件库。里面包含所有常用的native apr，比如工具类、hook相关等。
集成&问题排查
参考视频（39-42）
总结：可以学到gradle报错解决。CMake问题解决方法
组件库模块介绍：
adapter
存放一些常见的api，比如获取android 版本号，常见的一些全局变量放在这里面保存，比如JNIEnv
dlfc
dlfc本身实现
https://tech.meituan.com/2017/07/20/android-remote-debug.html
解决了，7.0以上dlopen无法打开私有库的问题。
elfUtils
主要是为了获取解决非导出函数失败的问题。
includes
保存了全部的nativeLib提供功能的头文件。放到一起是为了统一管理。
libpath
保存了常用的路径，比如不同版本的art路径，统一封装
mmvk
保存数据的工具，底层mmap实现。可以多进程共享数据，速度快。
raw_syscall
syscall的实现
utils->基础工具类
native开发常用的工具类
utils->hook工具类
utils->jnitrace（参考课程：45-46）
介绍：
监听目标So的jni调用
优势：
可以在不分析so的情况下，监听全部的jni调用，以此实现快速分析jni的目的。
utils->libc string 函数跟踪（参考课程：47）
介绍：
监听目标So的libc里面的字符串处理函数
优势：
可以在不分析so的情况下，监听全部的jni调用，以此实现快速分析字符串调用的目的。
utils->跟踪全部java调用（参考课程：48）
介绍：
监听全部的java方法调用并打印
优势：
监听java方法调用
总结：
自动化分析必备工具！！
BaseLib
功能：
公共Api库文件
常用的函数库：
Gson
将对象解析成字符串，字符串解析成对象
XposedAppium：
自动化点击框架，通过Xposed实现
内存漫游：
从内存中获取对象的实现
常用Utils
... ...
Test
测试项目App
BuildSrc
保存项目常用变量
常用的Hook库文件
Sandhook
Java代码Hook
Dobby
inlinehook库，主要是native hook
CMake编译学习
cmake和makefile的区别？
（参考视频：35）
cmake基础知识
cmake是android studio推荐的编译工具
cmake的构建方法不区分大小写
cmake是有执行顺序，从上到下执行，属于解释性脚本，存在自己的解释器
Cmake编译选项&参数学习
cmake_minimum_required
设置Cmake的版本号
SET
设置一个变量的值
include_directories
添加编译器路径，用于头文件的导入路径，防止存在找不到头文件的问题。
enable_language
启动asm，可以让cmake支持汇编指令的编译
string
字符串处理
https://blog.csdn.net/m0_57845572/article/details/118520561
add_library
添加一个库文件，可以是动态库，可以是静态库，取决于你是否是多个so，静态库可以作为依赖被动态库引入，。
add_executable
添加可执行文件，比如一个main程序，或者linux上面的一个命令。
find_library
查找一个lib库，用于链接，比如我们需要在A库里面引入B.so，就需要去查找B.so的路径
target_link_libraries
链接So，将已经编译好的.o或者.a文件进行连接
add_definitions
添加宏定义
add_subdirectory
添加一个编译路径，编译文件夹
Java转C++ 开发学习
指针&指针&引用&登记处
（参考课程：17）
面向对象&面向过程
狗 咬（狗，人）
狗 咬（人）
指针
auto *t = new struct test1();
int i1 = (*t).a;
int i2 = t->a;
函数指针
int (*t1)(string& str) = nullptr;
t1 = test;
引用
为什么要使用引用?
namespace
为什么要使用namespace
C++String的用法
static&extern
（参考课程：19）
static
static作用域介绍
extern
导入&导出介绍
C++字符串相关操作：
（参考课程18）
四种类型强转介绍
（参考课程：19）
const_cast
去除const强转
static_cast
对象和对象进行强转
reinterpret_cast
指针和指针之间强转
dynamic_cast（不常用）
对象和对象多肽之间强转
泛型
（参考课程：20）
存放类型限制
简介
不确定参数类型
参数
常见使用场景
集合类型限制
auto&推导decltype&namespace
（参考课程：21）
析构&构造
（参考课程：24）
保存类型的几种方式
（参考课程：25）
aotu
typedef
using
Class
（参考课程：23-24）
构造
析构
静态方法
集成
集合&容器
（参考课程：26-27）
常用容器介绍
vector
deque
list
array
string
map
set
采坑点介绍
智能指针
多层次释内存按常用框架
talloc介绍
case:
小张，小明，小王同住一个房子，他们的房子是他们的上下文。当房子被回收以后，所属的小张，小明，小王皆被回收。
常用的三种问题指针
强占用
std::unique_ptr
分享
std::shared_ptr
资源控制，不引用
std::weak_ptr
1）不要使用std::auto_ptr
2）当你需要一个独占资源所有权（访问权+生命控制权）的指针，且不允许任何外界访问，请使用std::unique_ptr
3）当你需要一个共享资源所有权（访问权+生命控制权）的指针，请使用std::shared_ptr
4）当你需要一个能访问资源，但不控制其生命周期的指针，请使用std::weak_ptr
推荐用法：一个shared_ptr和n个weak_ptr搭配使用.而不是n个shared_ptr。
逻辑上，大部分模型的生命在直观上总是受某一特定资源控制而不是多样东西共同控制。
程序上，能够完全避免生命周期互相控制引发的循环引用问题
总结
sr119020233
C++宏定义
（参考课程：29）
简介：
在编译期间做的事情
常见符号
#
将参数转换成字符串
##
参数链接
系统自带宏
__FILE__
文件名
__LINE__
行号
JNI&NDK相关知识
（参考课程：31-34）
简介
JNI Java和C的连接器，NDK是安卓提供的编译工具，可以直接在Native层调用安卓相关的C层API，比如在Java层去获取安卓版本号
调用Java
env常用方法介绍
作用域问题介绍
c调用c
调用方法细化
获取class
获取mid
调用对应类型的方法
jstring和char*互转问题
Java调用c
静态注册
java_com_example_test_MainActivity_inlinehook
java开头，包名，类名，方法名
动态注册
env->RegisterNatives
RegisterNatives
结论：
RegisterNatives 本质上是给so赋值的一个过程
注册流程分析
安卓9以下是将通过art_method里面的RegisterNative 进行注册，安卓12以上则是通过linker进行注册
自实现RegisterNatives
异常处理
简介
打印异常信息，当存在异常的情况
env->ExceptionDescribe();
检测是否存在异常
env->ExceptionCheck()
获取异常的对象&实例
env->ExceptionOccurred();
异常的消除clean
env->ExceptionClear();
注意：当存在异常的时候必须先clean在操作java方法。但是可以在java出异常以后操作C方法
如何打印具体的异常信息?
拿到异常实例以后，调用toString方法
jthrowable gjthrowable = env->ExceptionOccurred();
env->ExceptionClear();
jmethodID pjmethodId = env->GetMethodID(env->FindClass("java/lang/Object"), "toString",
"()Ljava/lang/String;");
jobject errorMsg = env->CallObjectMethod(pjthrowable, pjmethodId);