
看雪
SDC·2019
2019
安全开发者峰会
Security
Development
Conference

Android 0day漏洞检测沙箱系统的设计与实现

Moony Li of TrendMicro

Agenda

- 项目背景介绍
  - 漏洞攻防对抗
- 项目设计与实现
  - ✓ 整体思路
  - ✓ 关键模块选型与设计
    - ✓ APK Download
    - ✓ 静态分析与推理
    - ✓ 动态分析与推理
  - ✓ Misc
- Summary

微信: @flyic_t
https://github.com/silvermoonsecurity

Moony Li

- 乙方安全架构
- 9+年安全经验
- Windows、Mac安全沙箱开发
- Android，iOS等平台漏洞挖掘与利用
- 攻防对抗

国际会议：
BlackHat Europe 2016,
Black Hat Asia 2018,
Black Hat USA 2018 Arsenal,
Black Hat Europe 2018,
Blackhat USA 2019

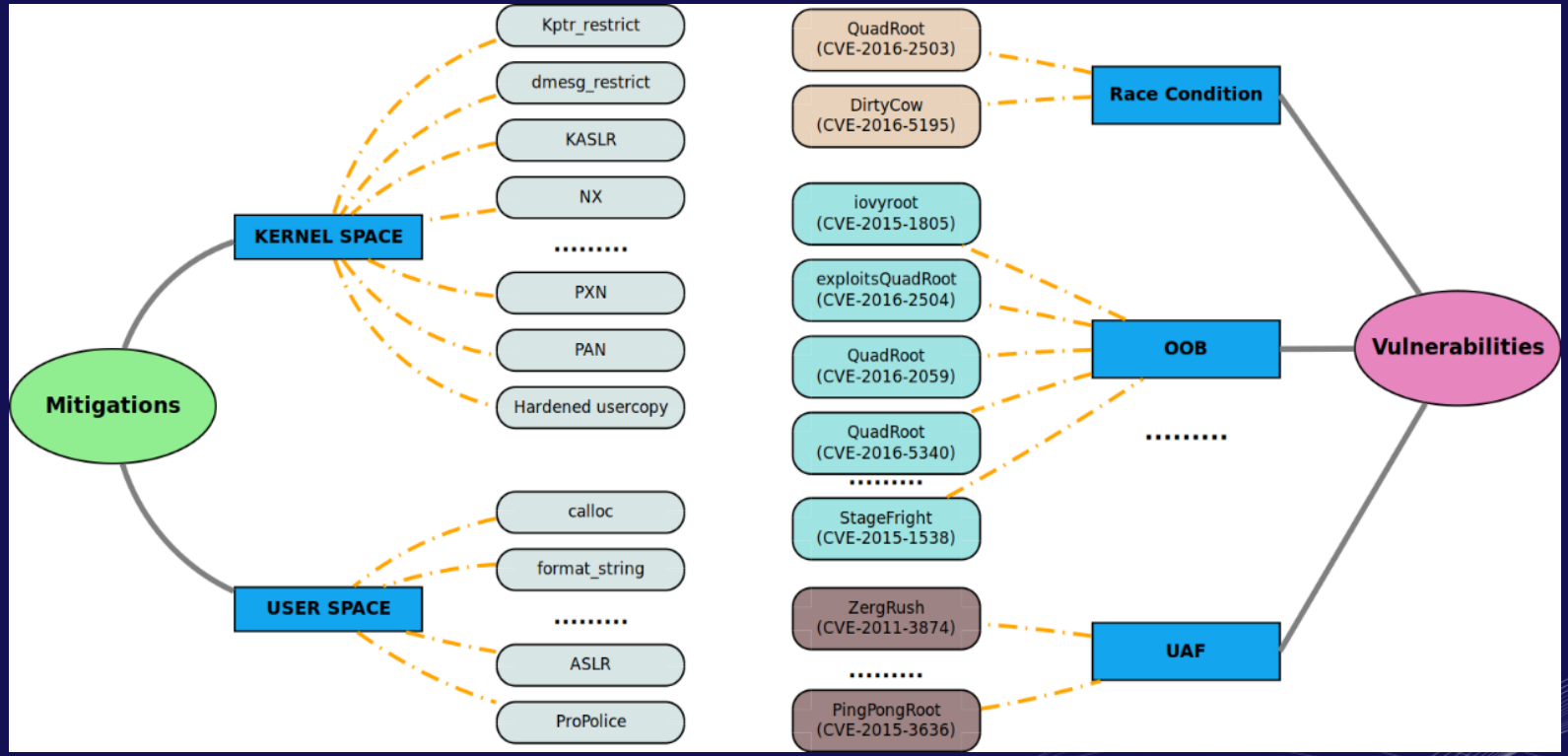
Android Mitigation演进 1/2

Evolution of Android Security										
Feature	1.5	2.3	4.0	4.1	4.2	4.3	4.4	5.0	6.0	7.0
ProPolice	√	√	√	√	√	√	√	√	√	√
Safe_iop	√	√	√	√	√	√	√	√	√	√
Calloc	√	√	√	√	√	√	√	√	√	√
Format-security		√	√	√	√	√	√	√	√	√
NX		√	√	√	√	√	√	√	√	√
Mmap_min_addr		√	√	√	√	√	√	√	√	√
ASLR			√	√	√	√	√	√	√	√
PIE				√	√	√	√	√	√	√
Dmesg_restrict				√	√	√	√	√	√	√
Kptr_restrict				√	√	√	√	√	√	√
Verify Apps					√	√	√	√	√	√
Premium SMS Control					√	√	√	√	√	√
Always-on VPN					√	√	√	√	√	√
Certificate Pinning					√	√	√	√	√	√
Installd hardening					√	√	√	√	√	√
Init script hardening					√	√	√	√	√	√
FORTIFY_SOURCE					√	√	√	√	√	√
ContentProvider default configuration					√	√	√	√	√	√
OpenSSL Cryptography improve					√	√	√	√	√	√
SELinux permissive						√	√	√	√	√
No setuid/setgid						√	√	√	√	√
ADB Authentication						√	√	√	√	√
Capability bounding						√	√	√	√	√
KeyStore&BoundKey						√	√	√	√	√
Text relocation protection						√	√	√	√	√
SElinux enforcing							√	√	√	√
Per User VPN							√	√	√	√
Full disk encryption								√	√	√
Smart Lock								√	√	√
Guest modes								√	√	√
WebView update without OTA								√	√	√
Runtime Permissions									√	√
Verified Boot									√	√
Hardware-Isolated Security									√	√
Fingerprints									√	√
Clear Text Traffic									√	√
USB Access Control:									√	√
										?

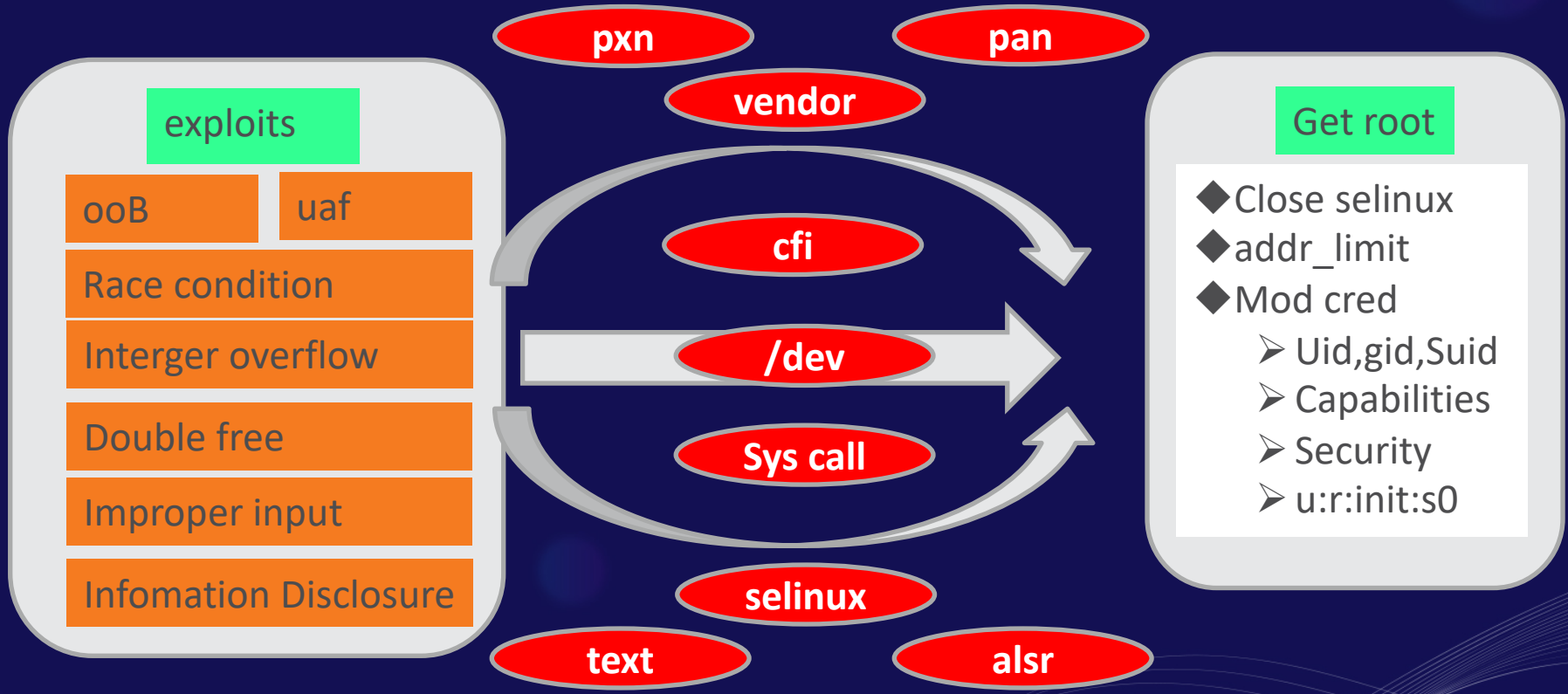
Android Mitigation演进 2/2

Android 5(<=)	PXN,Selinux
Android 6	Device-mapper-verity
Android 7	Integer Overflow Sanitization
Android 8	Post-init read-only memory,Hardened Usercopy,PAN,KASLR
Android 9	llvm-cfi,Integer Overflow Sanitization
Android Q(>=)	lld linker,protect more of CFI

Mitigation vs Vulnerability



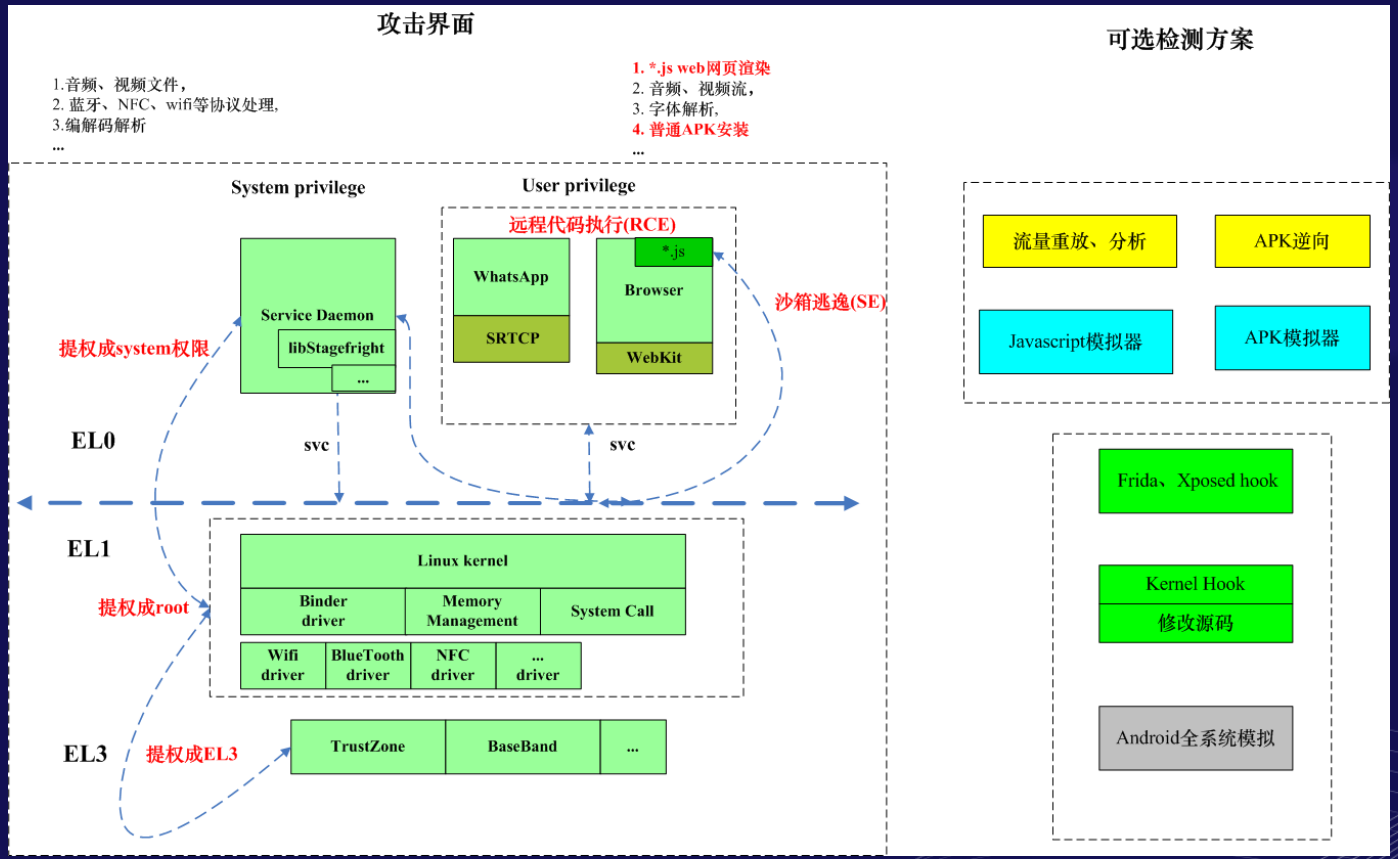
Migitation vs Exploit



Agenda

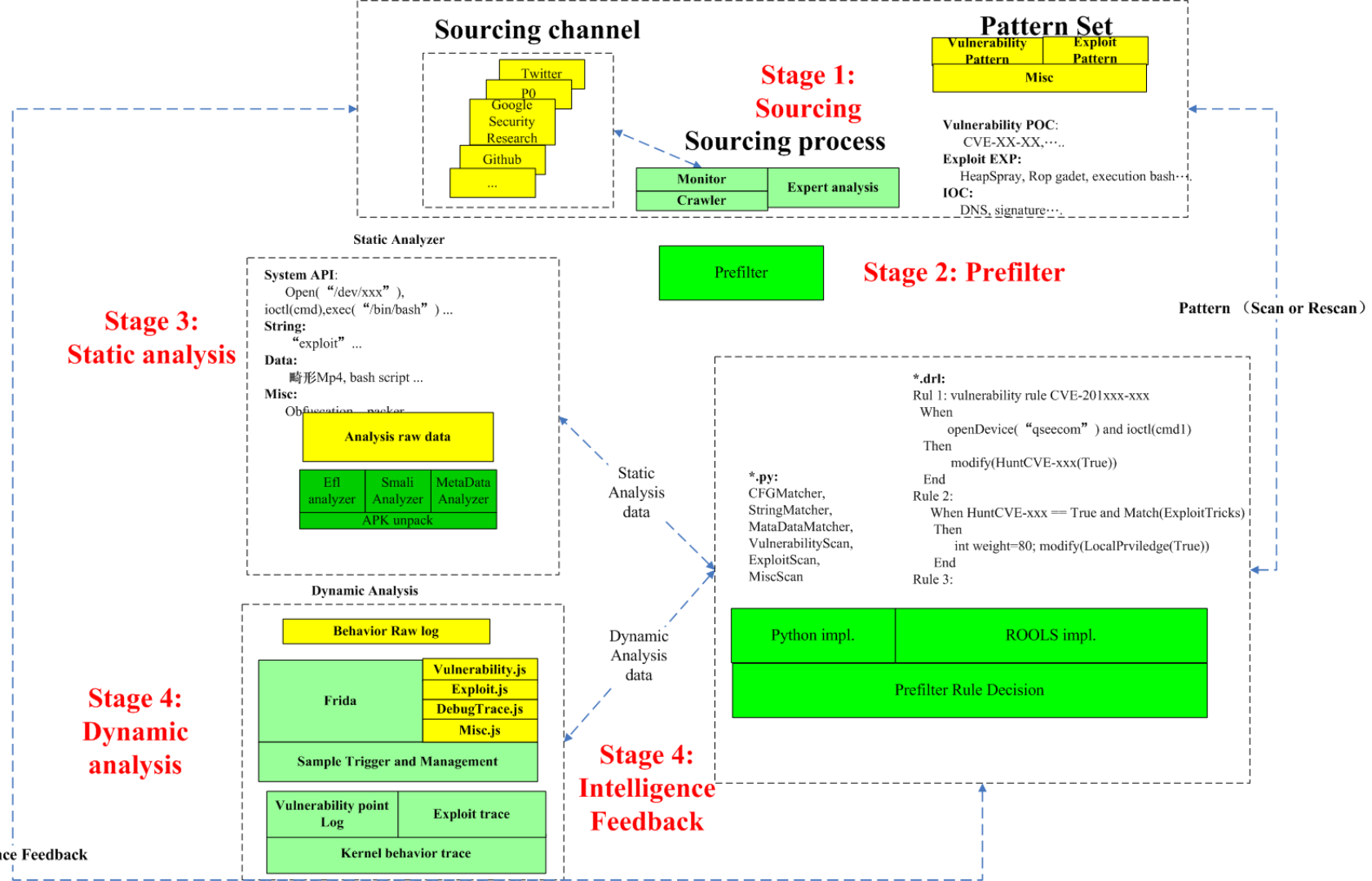
- 项目背景介绍
- 设计与实现
  - ✓ 整体思路
  - ✓ 关键模块选型与设计
    - ✓ APK Download
    - ✓ 静态分析与推理
    - ✓ 动态分析与推理
  - ✓ Misc
- Summary

知己知彼：攻击界面 VS 可选方案

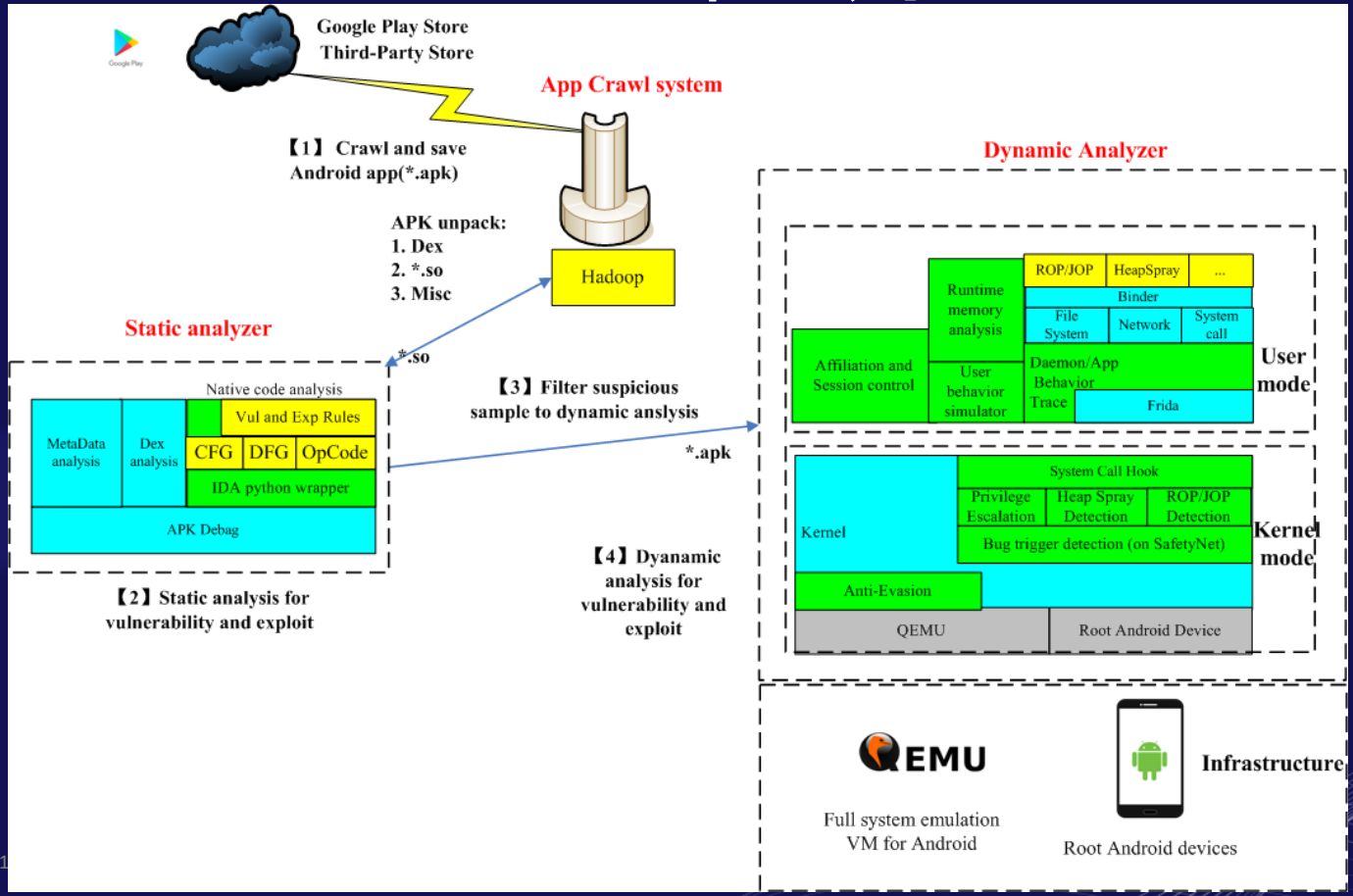


各方案点评

	动态分析	Emulation	静态分析
代码混淆，加壳，加花...	√	√	×
漏洞代码执行依赖（分阶段运行、反调试、虚拟机检测、UI检测、用户输入检测...）	×	√ ×	√
Download & execution	×	×	×
Misc			


Sourcing channel
Twitter
P0
Google Security Research
Github
...
Stage 1: Sourcing
Sourcing process
Monitor
Crawler
Expert analysis
Pattern Set
Vulnerability Pattern
Exploit Pattern
Misc
Vulnerability POC:
CVE-XX-XX,⋯..
Exploit EXP:
HeapSpray, Rop gadet, execution bash⋯.
IOC:
DNS, signature⋯.
Static Analyzer
Prefilter
Stage 2: Prefilter
Pattern （Scan or Rescan）
Stage 3: Static analysis
System API:
Open(“/dev/xxx”),
ioctl(cmd),exec(“/bin/bash”) ...
String:
“exploit” ...
Data:
畸形Mp4, bash script ...
Misc:
Obfuscation packer
Analysis raw data
Efl analyzer
Smali Analyzer
MetaData Analyzer
APK unpack
Static Analysis data
*.drl:
Rul 1: vulnerability rule CVE-201xxx-xxx
When
openDevice(“qseecom”) and ioctl(cmd1)
Then
modify(HuntCVE-xxx(True))
End
Rule 2:
When HuntCVE-xxx == True and Match(ExploitTricks)
Then
int weight=80; modify(LocalPrviledge(True))
End
Rule 3:
*.py:
CFGMatcher,
StringMatcher,
MataDataMatcher,
VulnerabilityScan,
ExploitScan,
MiscScan
Python impl.
ROOLS impl.
Prefilter Rule Decision
Dynamic Analysis
Behavior Raw log
Frida
Vulnerability.js
Exploit.js
DebugTrace.js
Misc.js
Sample Trigger and Management
Vulnerability point Log
Exploit trace
Kernel behavior trace
Dynamic Analysis data
Stage 4: Dynamic analysis
Stage 4: Intelligence Feedback
Intelligence Feedback

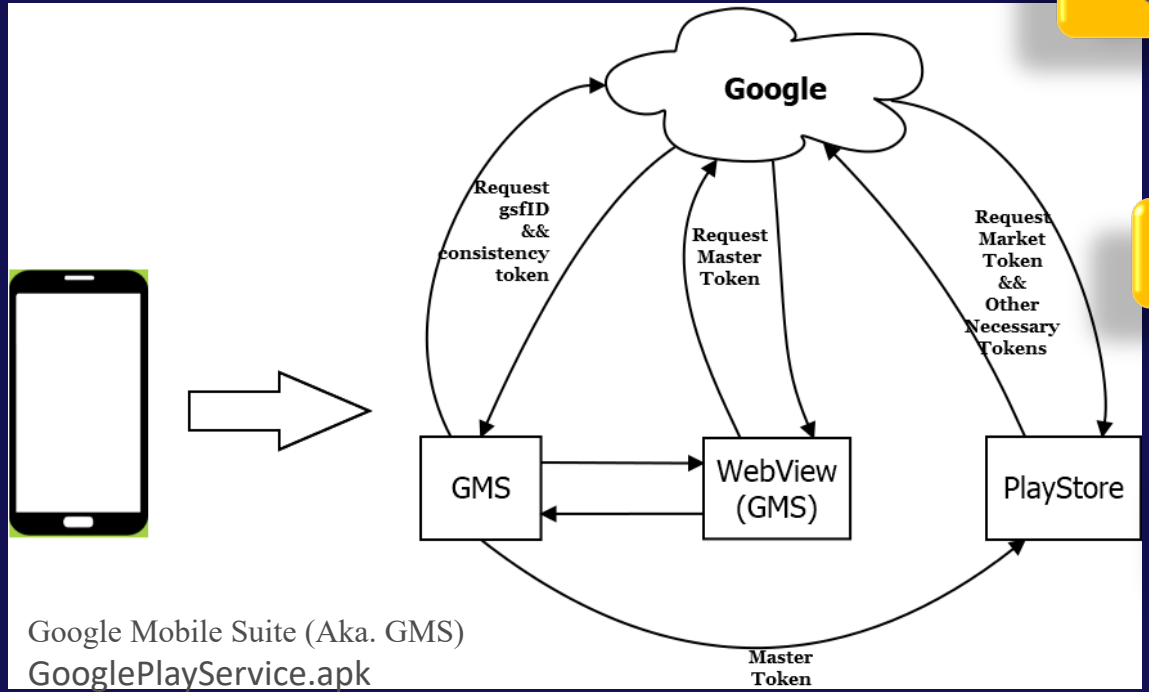
Solution Overview 内部视角



Agenda

- 项目背景介绍
- 设计与实现
  - ✓ 整体思路
  - ✓ 关键模块选型与设计
    - ✓ APK Download
    - ✓ 静态分析与推理
    - ✓ 动态分析与推理
  - ✓ Misc
- Summary

APK sourcing of Google Play



Google Mobile Suite (Aka. GMS)
GooglePlayService.apk

Check In

收集mac address, IMEI code, network type, sim ca
carrier information, hardware information来产生gsfID

Webview login

用户输入GP的账密，返回oAuth token作为 maste

Download

获取market token等下载安装APK

Tips

- 分布式部署
  - ✓不同地理，IP段，代理
  - ✓控制频次
- 设备模拟
  - ✓ Pixel, Nexus, Samsung, Moto …
  - ✓Xposed,Frida+真机
  - ✓定制ROM

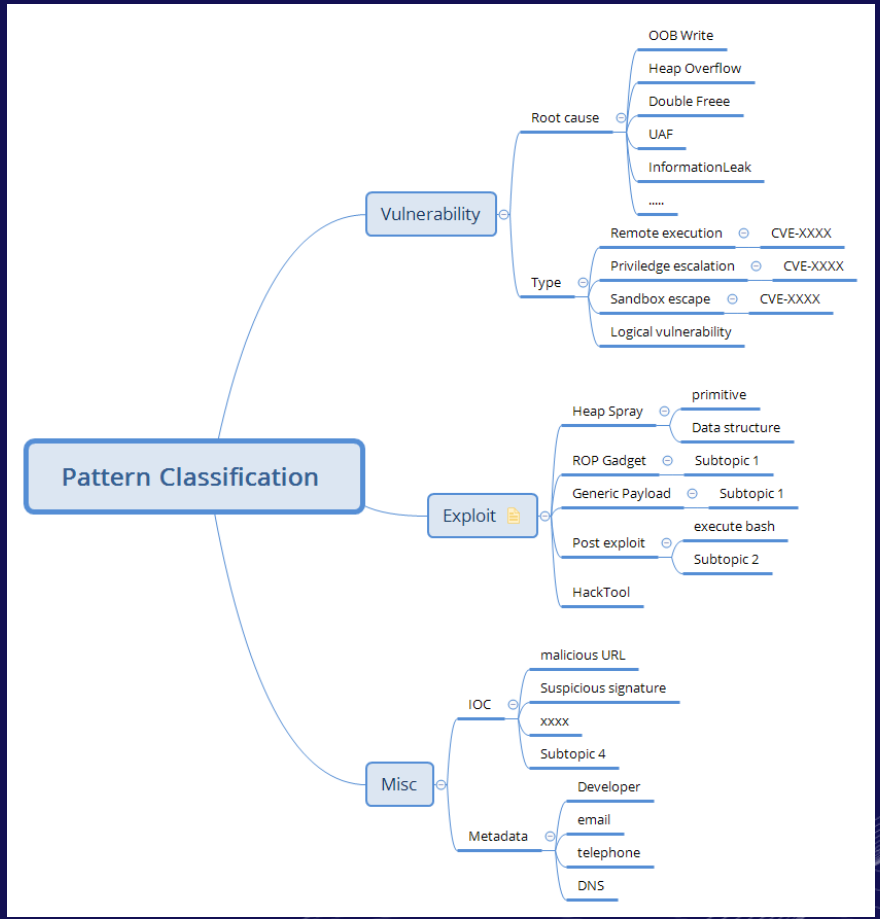
Number	Description
1	imei
2	androidId
3	digest
4	checkin
…	……
26	wIsInChina
27	LplWearableDeviceRelated
29	yFetchSystemUpdates

Agenda

- 项目背景介绍
- 设计与实现
  - ✓ 整体思路
  - ✓ 关键模块选型与设计
    - ✓ APK Download
    - ✓ 静态分析与推理
    - ✓ 动态分析与推理
  - ✓ Misc
- Summary

Pattern 收集

- N-Day 漏洞特征
- 0Day Exploit经验
- 辅助
  - ✓ Hack tool
  - ✓ Bash



静态分析方案总览

- Rough 模式（1.7w）
  - ✓ 字符串全匹配
- Pattern 模式（17）
  - Yara Rule
- 精准模式 (More for researcher)
  - ✓ 程序分析CFG，xref to， xref from
  - ✓ 伪代码
  - ✓ ...

DirtyCow in the wild

Cyberespionage Campaign Sphinx Goes Mobile With AnubisSpy

Posted on: December 19, 2017 at 4:07 am **Posted in:** Mo
Author: Mobile Threat Response Team

2

by Ecular Xu and Grey Guo

Android malware like ransomware exemplify how the p
can be lucrative for cybercriminals. But there are also
threats stirring up as of late: attacks that spy on and s
from specific targets, crossing over between desktop
mobile devices.

Take for instance several malicious apps we came
cyberespionage capabilities, which were targeting
speaking users or Middle Eastern countries. Thes
published on Google Play — but have since bee
and third-party app marketplaces. We named th
apps AnubisSpy (ANDROIDOS_ANUBISSPY)
watchdog.

We construe AnubisSpy to be linked to the cy
shared file structures and command-and-cor
that while AnubisSpy's operators may also
campaigns.

ZNIU: First Android Malware to Exploit Dirty COW Vulnerability

Posted on: September 25, 2017 at 5:00 am
Posted in: Bad Sites, Malware, Mobile, Vulnerabilities **Author:** Mobile Threat Response Team

29

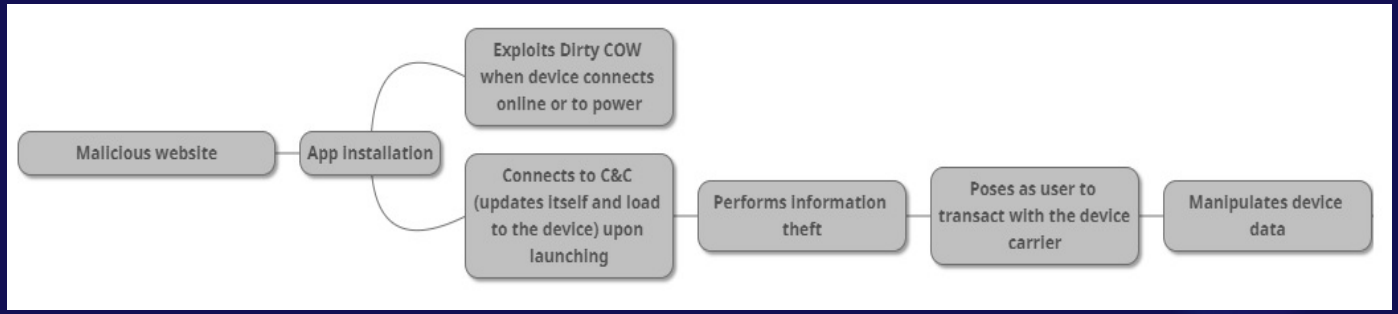
By Jason Gu, Veo Zhang, and Seven Shen

We have disclosed this security issue to Google, who verified that they have protections in place against ZNIU courtesy of Google Play Protect.

The Linux vulnerability called Dirty COW (CVE-2016-5195) was first disclosed to the public in 2016. The vulnerability was discovered in upstream Linux platforms such as Redhat, and Android, which kernel is based on Linux. It was categorized as a serious privilege escalation flaw that allows an attacker to gain root access on the targeted system. Dirty COW attacks on Android has been silent since its discovery, perhaps because it took attackers some time to build a stable exploit for major devices. Almost a year later, Trend Micro researchers captured samples of ZNIU (detected as AndroidOS_ZNIU)—the first malware family to exploit the vulnerability on the Android platform.

The ZNIU malware was detected in more than 40 countries last month, with the majority of the victims found in China and India. We also detected the malware in the U.S., Japan, Canada, Germany, and Indonesia. As of this writing, we have detected more than 5,000 affected users. Our data also shows that more than 1,200 malicious apps that carry ZNIU were found in malicious websites with an existing rootkit that exploits Dirty COW, disguising themselves as pornography and game apps, among others.

In the Wild Exploit Hunt



- 300,000 malicious apps that carry the ZNIU malware in the wild by September 27, 2017
- Appears as a porn app

DirtyCow提权漏洞POC特征

- dirtycow
  - ✓ mmap
  - ✓ * Madvise, 4
  - ✓ pthread_create
  - ✓ pthread_join
  - ✓ Open

```
v6 = mmap(0LL, 4096LL, 3LL, 33LL, 0xFFFFFFFFLL, 0LL);
v7 = fork(v6);
v8 = v7;
if ( (v7 & 0x80000000) != 0 )
{
  perror("fork:0x1 root error:");
  exit(0LL);
}
if ( !v7 )
  goto LABEL_15;
sprintf(&v14, "/proc/%d/mem", v7);
v9 = open(&v14, 2LL);
if ( v9 == -1 )
  printf("open");
v10 = 0x100000;
do
{
  lseek(v9, v4, 0LL);
  write(v9, v3, v5);
  --v10;
}
while ( v10 );
kill(v8, 10LL);
wait(&v13);
printf("Parent is over..status == %d\n");
close(v9);
result = _stack_chk_guard;
if ( v24 != _stack_chk_guard )
{
LABEL_15:
  v12 = 0x100000;
  do
  {
    madvise(v4, v5, 4);
    --v12;
  }
  while ( v12 );
  exit(0LL);
```

Exploit of DirtyCOW

- Race condition in copy-on-write

- Main pattern:
  - "madvise" system must be contained in user mode
- Affiliation pattern:
  - map memory (e.g. mmap), multiple thread/process (e.g. fork, pthread_create) are optional.

```
v6 = mmap(0LL, 4096LL, 3LL, 33LL, 0xFFFFFFFFLL, 0LL);
v7 = fork(v6);
v8 = v7;
if ( (v7 & 0x80000000) != 0 )
{
  perror("fork:0x1 root error:");
  exit(0LL);
}
if ( !v7 )
  goto LABEL_15;
sprintf(&v14, "/proc/%d/mem", v7);
v9 = open(&v14, 2LL);
if ( v9 == -1 )
  printf("open");
v10 = 0x100000;
do
{
  lseek(v9, v4, 0LL);
  write(v9, v3, v5);
  --v10;
}
while ( v10 );
kill(v8, 10LL);
wait(&v13);
printf("Parent is over..status == %d\n");
close(v9);
result = _stack_chk_guard;
if ( v24 != _stack_chk_guard )
{
ABEL_15:
  v12 = 0x100000;
  do
  {
    madvise(v4, v5, 4);
    --v12;
  }
  while ( v12 );
  exit(0LL);
```

Yara Rule

CVE-2016-5195.yar

```
rule Linux_DirtyCow_Exploit {
   meta:
      description = "Detects Linux Dirty Cow Exploit - CVE-2012-0056 and CVE-2016-5195"
      author = "Florian Roth"
      reference = "http://dirtycow.ninja/"
      date = "2016-10-21"
   strings:
      $a1 = { 48 89 D6 41 B9 00 00 00 00 41 89 C0 B9 02 00 00 00 BA 01 00 00 00 BF 00 00 00 00 }
      $b1 = { E8 ?? FC FF FF 48 8B 45 E8 BE 00 00 00 00 48 89 C7 E8 ?? FC FF FF 48 8B 45 F0 BE 00 00 00 00 48 89 }
      $b2 = { E8 ?? FC FF FF B8 00 00 00 00 }

      $source1 = "madvise(map,100,MADV_DONTNEED);"
      $source2 = "=open(\"/proc/self/mem\",O_RDWR);"
      $source3 = ",map,SEEK_SET);"

      $source_printf1 = "mmap %x"
      $source_printf2 = "procselfmem %d"
      $source_printf3 = "madvise %d"
      $source_printf4 = "[-] failed to patch payload"
      $source_printf5 = "[-] failed to win race condition..."
      $source_printf6 = "[*] waiting for reverse connect shell..."

      $s1 = "/proc/self/mem"
      $s2 = "/proc/%d/mem"
      $s3 = "/proc/self/map"
      $s4 = "/proc/%d/map"

      $p1 = "pthread_create" fullword ascii
      $p2 = "pthread_join" fullword ascii
   condition:
      ( uint16(0) == 0x457f and $a1 )
      /*0     byte       0x457f        "executable: This file is an ELF file."*/
      or
      all of ($b*) or
      3 of ($source*) or
      ( uint16(0) == 0x457f and 1 of ($s*) and all of ($p*) and filesize < 20KB )

}
```

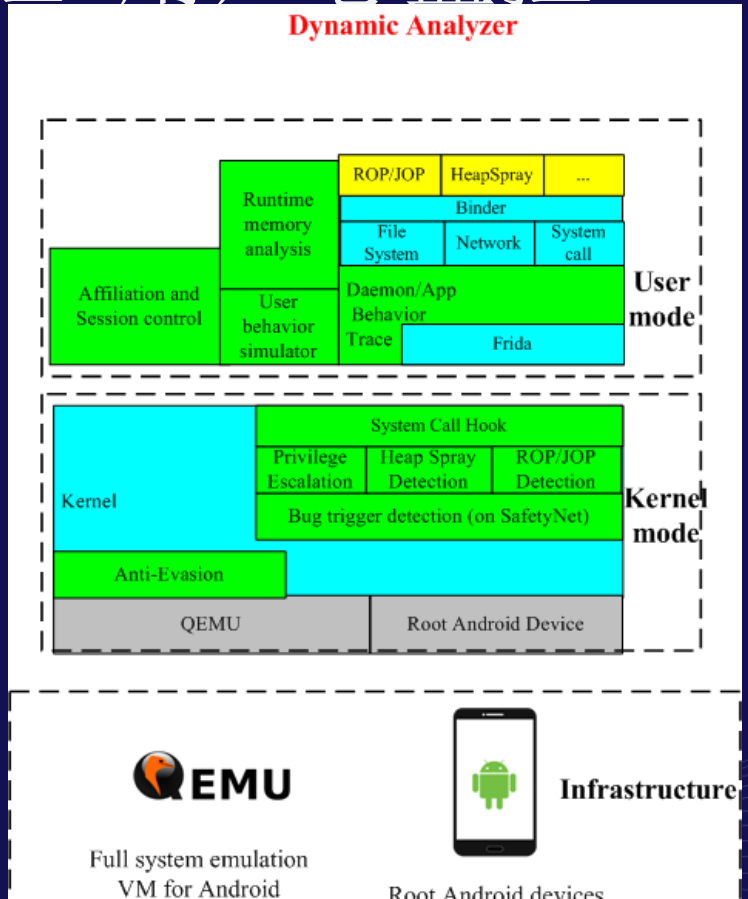
Agenda

- 项目背景介绍
- 设计与实现
  - ✓ 整体思路
  - ✓ 关键模块选型与设计
    - ✓ APK Download
    - ✓ 静态分析与推理
    - ✓ 动态分析与推理
  - ✓ Misc
- Summary

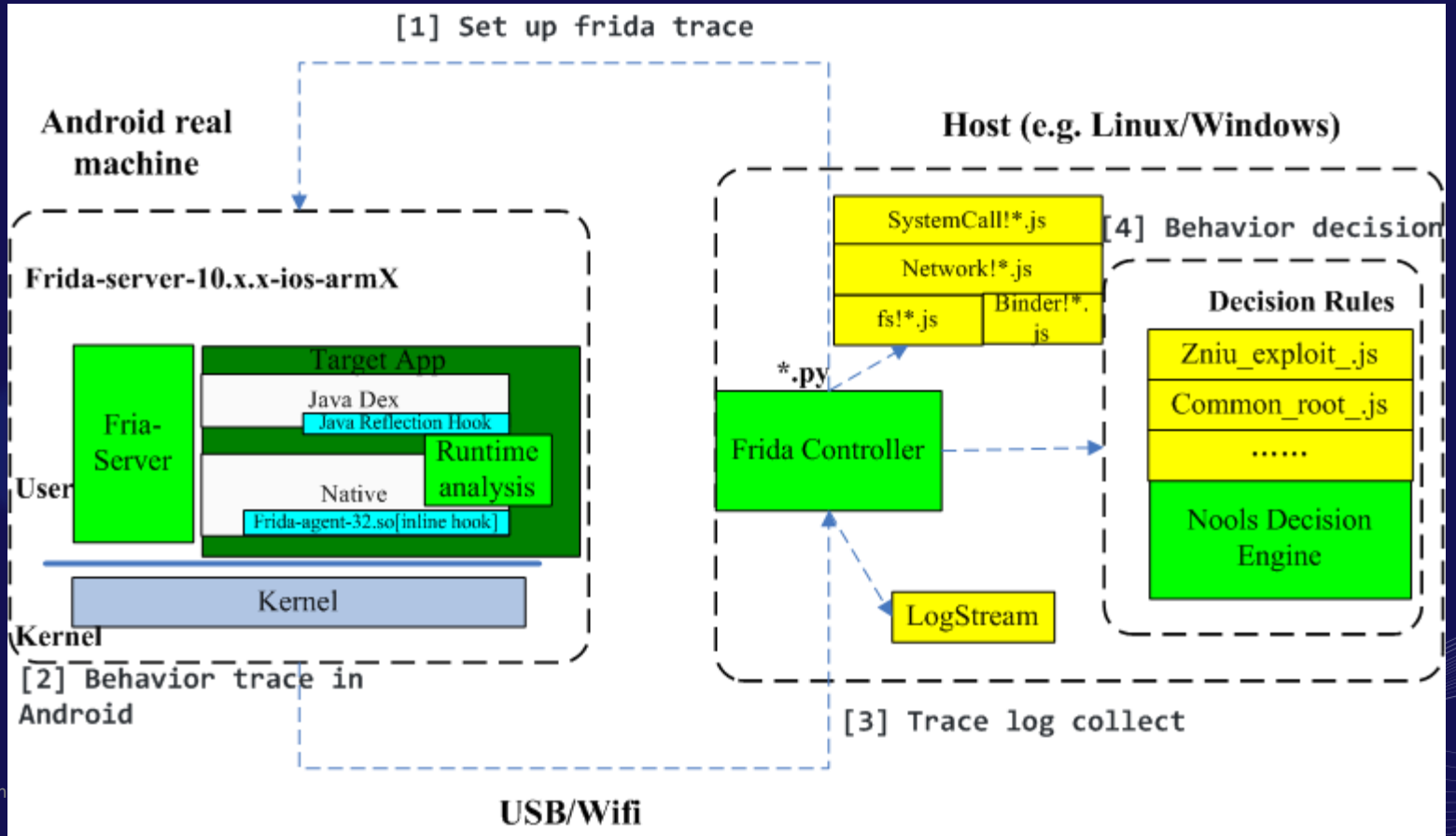
Module Components

- 1. Kernel Mode Detection
  - ✓Focus on kernel privilege escalation
  - ✓Such as UAF, double free, oob,…
- 2. User Mode Detection
  - ✓Focus on sandbox escape
  - ✓Such as RCE, mediaserverd vulnerabilities

内核态监控+用户态监控



Frida internal



内核态监控策略

- SaftyNet
- Heap Spray detection
- ROP detection
- Privilege Escalation detection

内核态监控策略

SaftyNet

Heap Spray detection

ROP detection

Privilege Escalation detection

SaftyNet - CVE-2016-0846 for example

```
Sanity check IMemory access versus underlying mmap

Bug 26877992

Change-Id: Ibbf4b106le4675e4e96bc944a865b53eaf6984fe

diff --git a/libs/binder/IMemory.cpp b/libs/binder/IMemory.cpp
index d8ed995..b9a8bce 100644
--- a/libs/binder/IMemory.cpp
+++ b/libs/binder/IMemory.cpp

@@ -26,6 +26,7 @@
 #include <sys/mman.h>

 #include <binder/IMemory.h>
+#include <cutils/log.h>
 #include <utils/KeyedVector.h>
 #include <utils/threads.h>
 #include <utils/Atomic.h>
@@ -187,15 +188,26 @@
             if (heap != 0) {
                 mHeap = interface_cast<IMemoryHeap>(heap);
                 if (mHeap != 0) {
-                    mOffset = o;
-                    mSize = s;
+                    size_t heapSize = mHeap->getSize();
+                    if (s <= heapSize
+                            && o >= 0
+                            && (static_cast<size_t>(o) <= heapSize - s)) {
                         mOffset = o;
                         mSize = s;
+                    } else {
+                        // Hm.
+                        android_errorWriteWithInfoLog(0x534e4554,
+                            "26877992" -1, NULL, 0);
+                        mOffset = 0;
+                        mSize = 0;
+                    }
                 }
             }
```

Google SafetyNet Exploit detection log

```
04-25 17:01:50.372   873  3214 I am_proc_start: [0,3243,10086,com.example.cve20160846,activity,com.example.cve20160846/.MainActivity]
04-25 17:01:50.403   873  3214 I am_proc_bound: [0,3243,com.example.cve20160846]

04-25 17:01:50.405   873  3214 I am_restart_activity: [0,124732844,601,com.example.cve20160846/.MainActivity]
04-25 17:01:50.544   484  2823 I snet_event_log: [26877992,-1,]
04-25 17:01:50.548  3243  3243 I am_on_resume_called: [0,com.example.cve20160846.MainActivity]
04-25 17:01:50.550   873  3934 I force_gc: Binder
04-25 17:01:50.653   873   926 I am_activity_launch_time: [0,124732844,com.example.cve20160846/.MainActivity,297,297]
```

Strategy of Kernel Mode detection

SaftyNet

Heap Spray detection

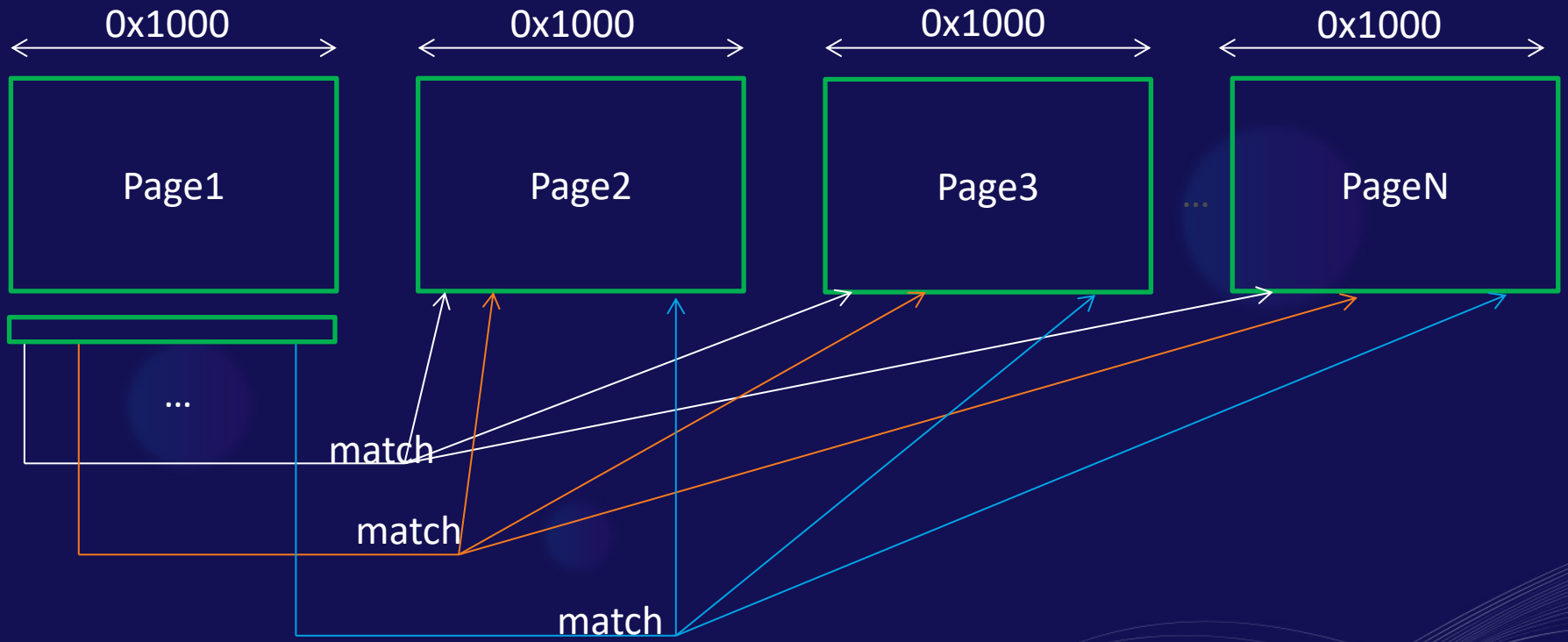
ROP detection

Privilege Escalation detection

Heap Spray Features

- I.Allocation：
  - malloc/kmalloc/vmalloc/new/mmap…
- II.Copy/Mapp:
  - memcpy/physmap…
- III. Socket
  - sendmsg/sendmmsg/…

HeapSpray检测逻辑



Return when find x match

Heap Spray 检测模式

- Runtime Allocation模式
  - Hook kmalloc、特殊结构malloc的特殊内核API等，计算调用的频次、每次开辟的大小、连续开辟的大小是否超越上限
- 状态监控模式
  - 实时监控内核空间的Page、slab等内存Pattern

DABox - CVE-2015-1538 detection(1/3)

```
02D0h: 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
02E0h: 00 1E C0 08 74 78 33 67 20 00 D0 41 18 00 D0 41  ..À.tx3g .ÐA..ÐA
02F0h: 01 00 00 00 AD DB DE C0 28 00 D0 41 10 00 00 00  ....-ÛÞÀ(.ÐA....
0300h: 30 00 D0 41 BE BA 0D F0 00 00 DE C0 04 00 DE C0  0.ÐA¾º.ð..ÞÀ..ÞÀ
0310h: 08 00 DE C0 50 28 00 B0 30 00 F0 F0 50 00 D0 41  ..ÞÀP(.°0.ððP.ÐA
0320h: 98 2A 00 B0 B3 38 00 B0 00 00 D0 41 00 10 00 00  ˜*.°³8.°..ÐA....
0330h: 07 00 00 00 03 D0 00 D0 04 D0 00 D0 44 11 00 B0  .....Ð.Ð.Ð.ÐD..°
0340h: 90 00 D0 41 5C 00 F0 F0 60 00 F0 F0 64 00 F0 F0  ..ÐA\.ðð`.ððd.ðð
```

Output

Address	Value
Found 492 occurrences of '07 00 00 00 03 d0'.	
330h	07 00 00 00 03 d0
1330h	07 00 00 00 03 d0
2330h	07 00 00 00 03 d0
3330h	07 00 00 00 03 d0
4330h	07 00 00 00 03 d0
5330h	07 00 00 00 03 d0
6330h	07 00 00 00 03 d0
7330h	07 00 00 00 03 d0
8330h	07 00 00 00 03 d0
9330h	07 00 00 00 03 d0
A330h	07 00 00 00 03 d0
B330h	07 00 00 00 03 d0
C330h	07 00 00 00 03 d0
D330h	07 00 00 00 03 d0
E330h	07 00 00 00 03 d0
F330h	07 00 00 00 03 d0
10330h	07 00 00 00 03 d0

DABox - CVE-2015-1538 detection(2/3)

- Intercept the buffer allocated
- Check the buffer contents through the algorithm
- Check the match results whether hit the threshold

DABox - CVE-2015-1538 detection(3/3)

- Detection results

```
Text
Heap Spray Detected! Duplicated binary data:
  0x7
  0x0
  0x0
  0x0
  0x3
  0xd0
  0x0
  0xd0
  0x4
  0xd0
  0x0
  0xd0
  0x44
  0x11
  0x0
  0xb0
Duplicated binary data address:
  0xb1d40050
  0xb1d41050
  0xb1d42050
  0xb1d43050
  0xb1d44050
Exploit detected:CVE-2015-1538!heap size:0x1ec008
```

Strategy of Kernel Mode detection

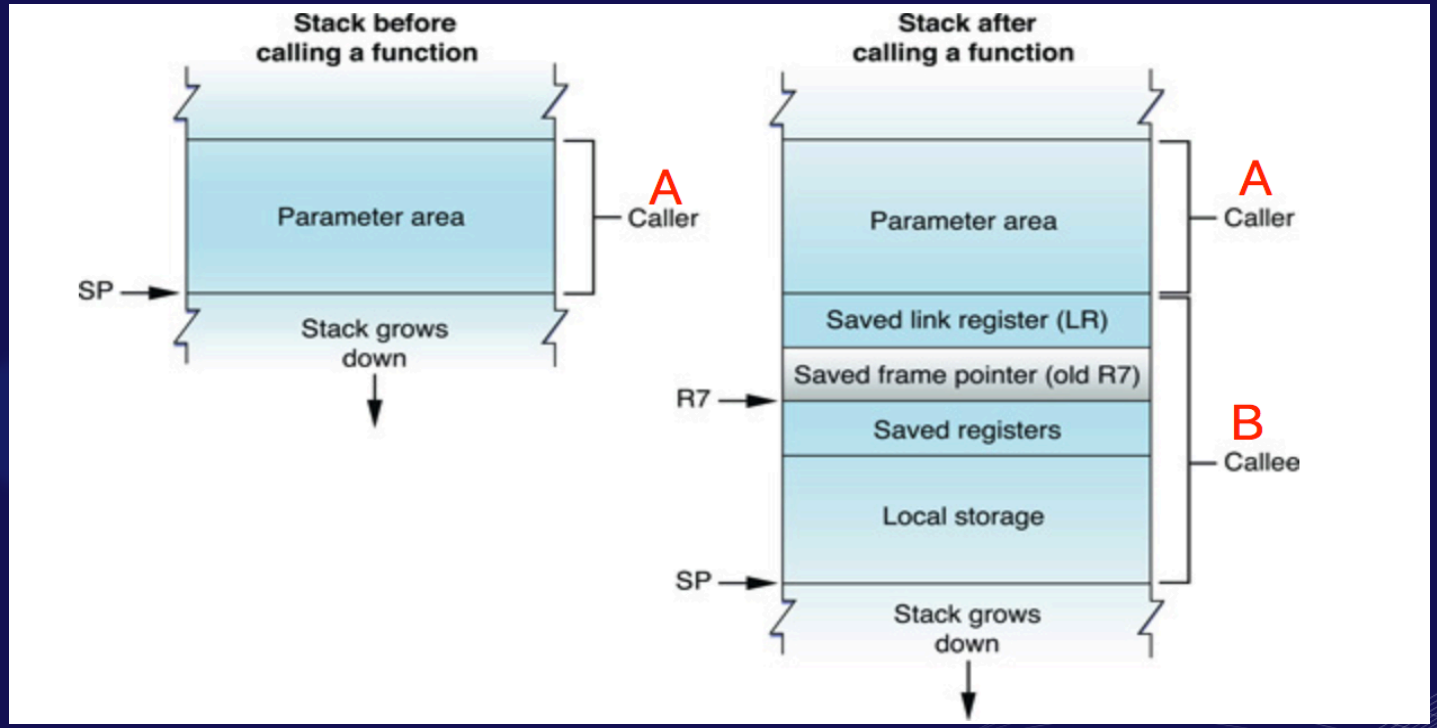
SaftyNet

Heap Spray detection

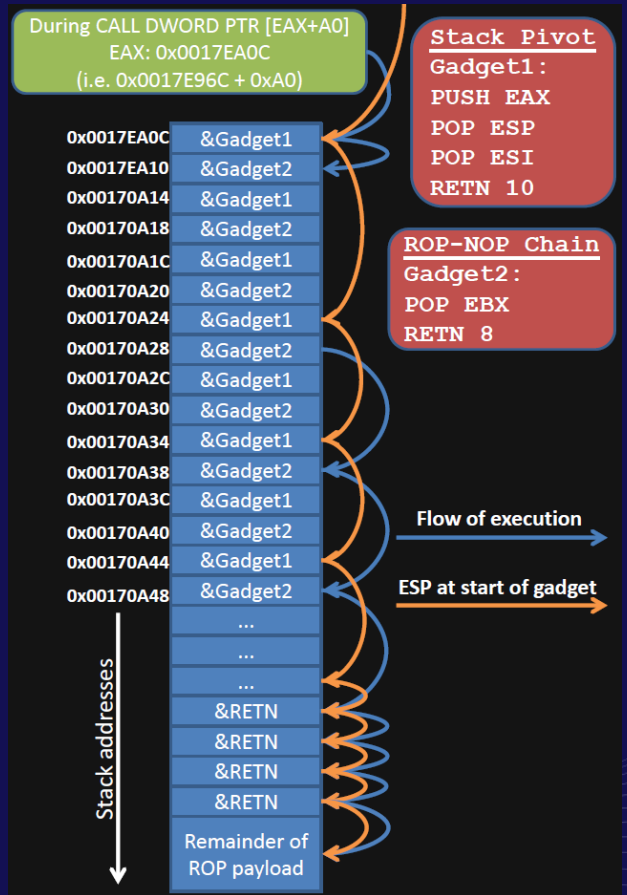
ROP detection

Privilege Escalation detection

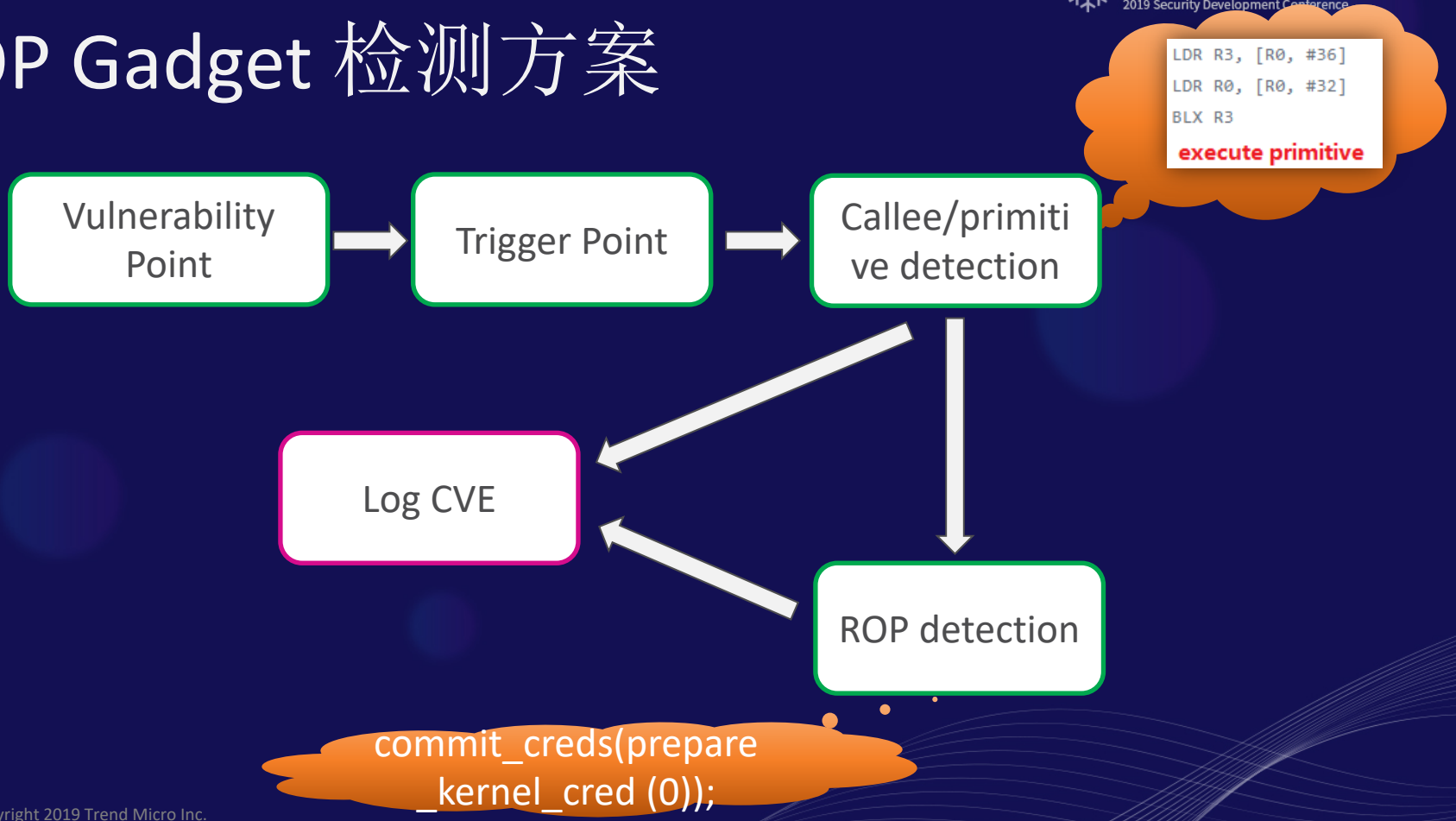
Arm Call Stack



ROP Gadget



ROP Gadget 检测方案



Strategy of Kernel Mode detection

SaftyNet

Heap Spray detection

ROP detection

Privilege Escalation detection

Privilege Escalation 检测方案

- monitor the target process status after:
  - ✓Process creation (“exec”)
  - ✓Shell command execution (“/bin/bash xxx.sh”)
  - ✓System file modification (inotify)
  - ✓ Logcat |grep “root successfully”:)
  - ✓…

CVE-2017-13315 Report

- Detection results

```
bash-4.2# cat 21EB716E6608A181317946087D62C68E7B4C79A7A00078F889DE85383B63E762.j
son
{"engine": "", "errorCode": 0, "result": 1, "endtime": "1562667999780", "comment
s": "test", "confirmed": 1, "pattern": ["CVE-2017-13315"], "virusName": "test",
"rule": "test", "begintime": "1562666928694", "sha256": "21EB716E6608A1813179460
87D62C68E7B4C79A7A00078F889DE85383B63E762"}bash-4.2#
bash-4.2#
bash-4.2#
bash-4.2#
```

Agenda

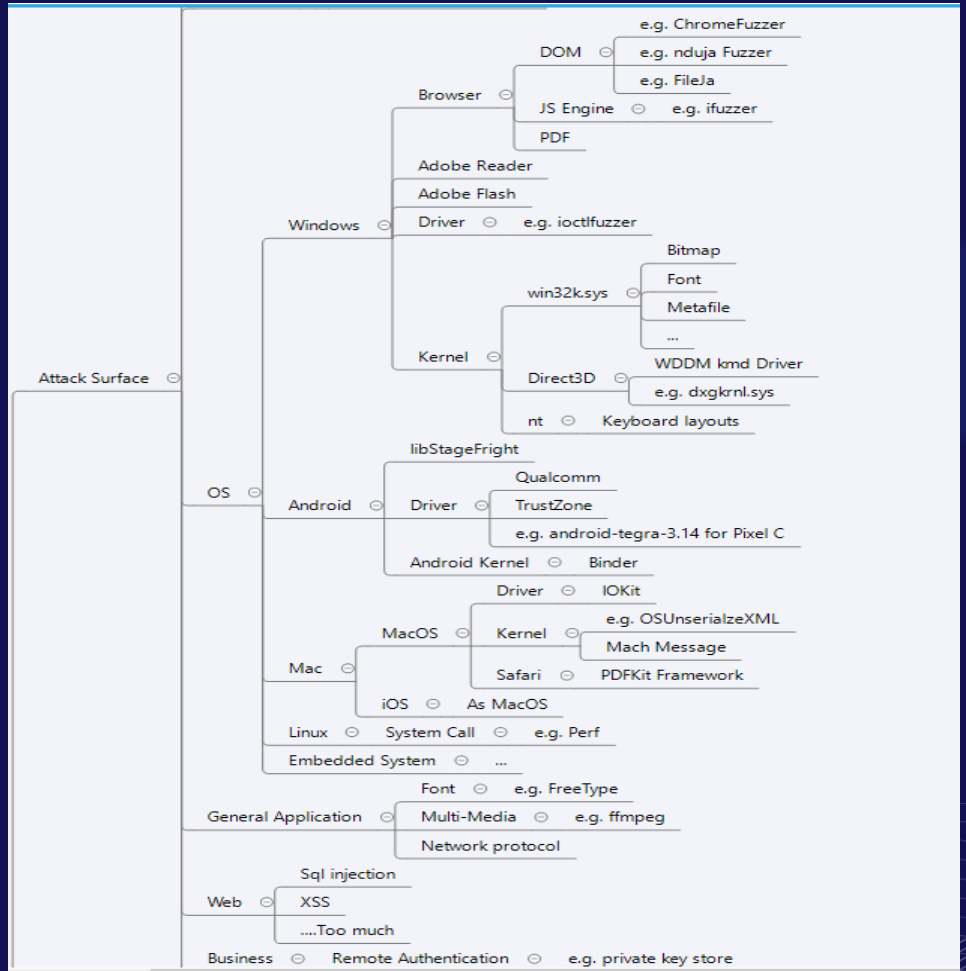
- 项目背景介绍
- 设计与实现
  - ✓ 整体思路
  - ✓ 关键模块选型与设计
    - ✓ APK Download
    - ✓ 静态分析与推理
    - ✓ 动态分析与推理
  - ✓ Misc
- Summary

Summary

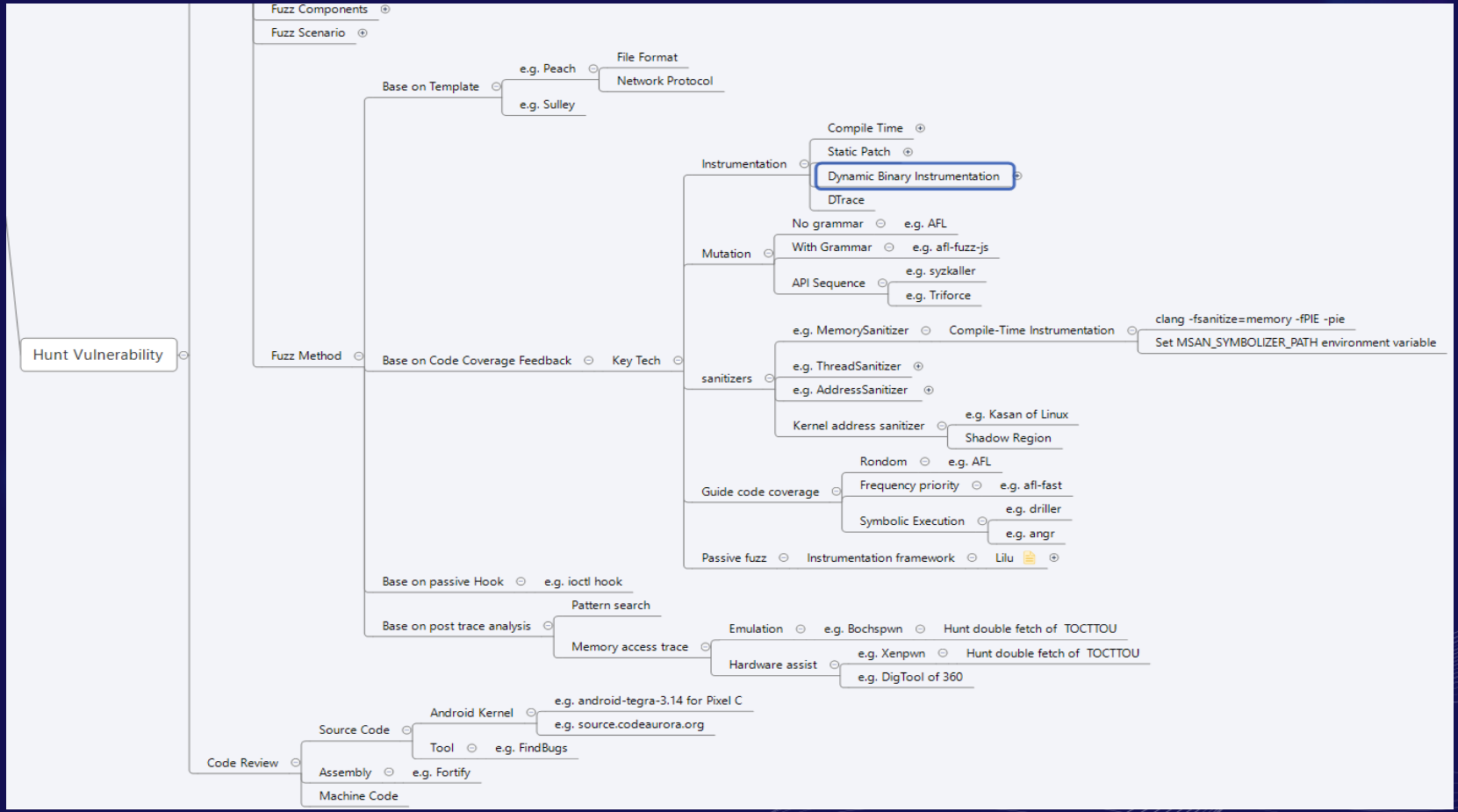
- for(;防<攻;防++)

  for(;攻<防;攻++)
- 未知攻，焉知防
- 专家经验->半自动化辅助->自动化？->Feedback

攻击界面


Attack Surface
OS
Windows
Browser
DOM
e.g. ChromeFuzzer
e.g. nduja Fuzzer
e.g. FileJa
JS Engine
e.g. ifuzzer
PDF
Adobe Reader
Adobe Flash
Driver
e.g. ioctlfuzzer
Kernel
win32k.sys
Bitmap
Font
Metafile
...
Direct3D
WDDM kmd Driver
e.g. dxgkrnl.sys
nt
Keyboard layouts
Android
libStageFright
Driver
Qualcomm
TrustZone
e.g. android-tegra-3.14 for Pixel C
Android Kernel
Binder
Mac
MacOS
Driver
IOKit
Kernel
e.g. OSUnserialzeXML
Mach Message
Safari
PDFKit Framework
iOS
As MacOS
Linux
System Call
e.g. Perf
Embedded System
...
General Application
Font
e.g. FreeType
Multi-Media
e.g. ffmpeg
Network protocol
Web
Sql injection
XSS
....Too much
Business
Remote Authentication
e.g. private key store

漏洞挖掘


Hunt Vulnerability
Fuzz Method
Fuzz Components
Fuzz Scenario
Base on Template
e.g. Peach
File Format
Network Protocol
e.g. Sulley
Base on Code Coverage Feedback
Key Tech
Instrumentation
Compile Time
Static Patch
Dynamic Binary Instrumentation
DTrace
Mutation
No grammar
e.g. AFL
With Grammar
e.g. afl-fuzz-js
API Sequence
e.g. syzkaller
e.g. Triforce
sanitizers
e.g. MemorySanitizer
Compile-Time Instrumentation
clang -fsanitize=memory -fPIE -pie
Set MSAN_SYMBOLIZER_PATH environment variable
e.g. ThreadSanitizer
e.g. AddressSanitizer
Kernel address sanitizer
e.g. Kasan of Linux
Shadow Region
Guide code coverage
Rondom
e.g. AFL
Frequency priority
e.g. afl-fast
Symbolic Execution
e.g. driller
e.g. angr
Passive fuzz
Instrumentation framework
Lilu
Base on passive Hook
e.g. ioctl hook
Base on post trace analysis
Pattern search
Memory access trace
Emulation
e.g. Bochspwn
Hunt double fetch of TOCTTOU
Hardware assist
e.g. Xenpwn
Hunt double fetch of TOCTTOU
e.g. DigTool of 360
Code Review
Source Code
Android Kernel
e.g. android-tegra-3.14 for Pixel C
e.g. source.codeaurora.org
Tool
e.g. FindBugs
Assembly
e.g. Fortify
Machine Code

Q&A

微信: @flyic_t

https://github.com/silvermoonsecurity

Moony Li

- 乙方安全架构
- 9+年安全经验
- Windows、Mac安全沙箱开发
- Android，iOS等平台漏洞挖掘与利用
- 攻防对抗

国际会议：

BlackHat Europe 2016,
Black Hat Asia 2018,
Black Hat USA 2018 Arsenal,
Black Hat Europe 2018,
Blackhat USA 2019