安卓系统安全简介

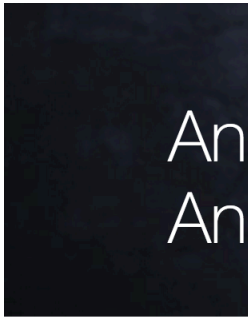
周亚金
http://yajin.org

关于我

- 北卡州立大学获得博士学位
- 从事移动安全研究6年+的经验
- 成果发表在国际知名学术安全会议
  - NDSS, IEEE S&P, USENIX SECURITY, CCS

智能手机越来越流行

手机安全事件层出不穷



Android 'smishing' vulnerability discovered by NCSU researchers; Google has a fix incoming

by Darren Murph | |

The Blu
that all
legitima
implica
affect a
applica

While t
an ente
HTC, S
AnyCo

The art of smishing (SMS
the wizards at NC State
aforesaid act back into t
hole and confirmed that
simply, if an Android use

Exploiting the Futex Bug and uncovering Towelroot

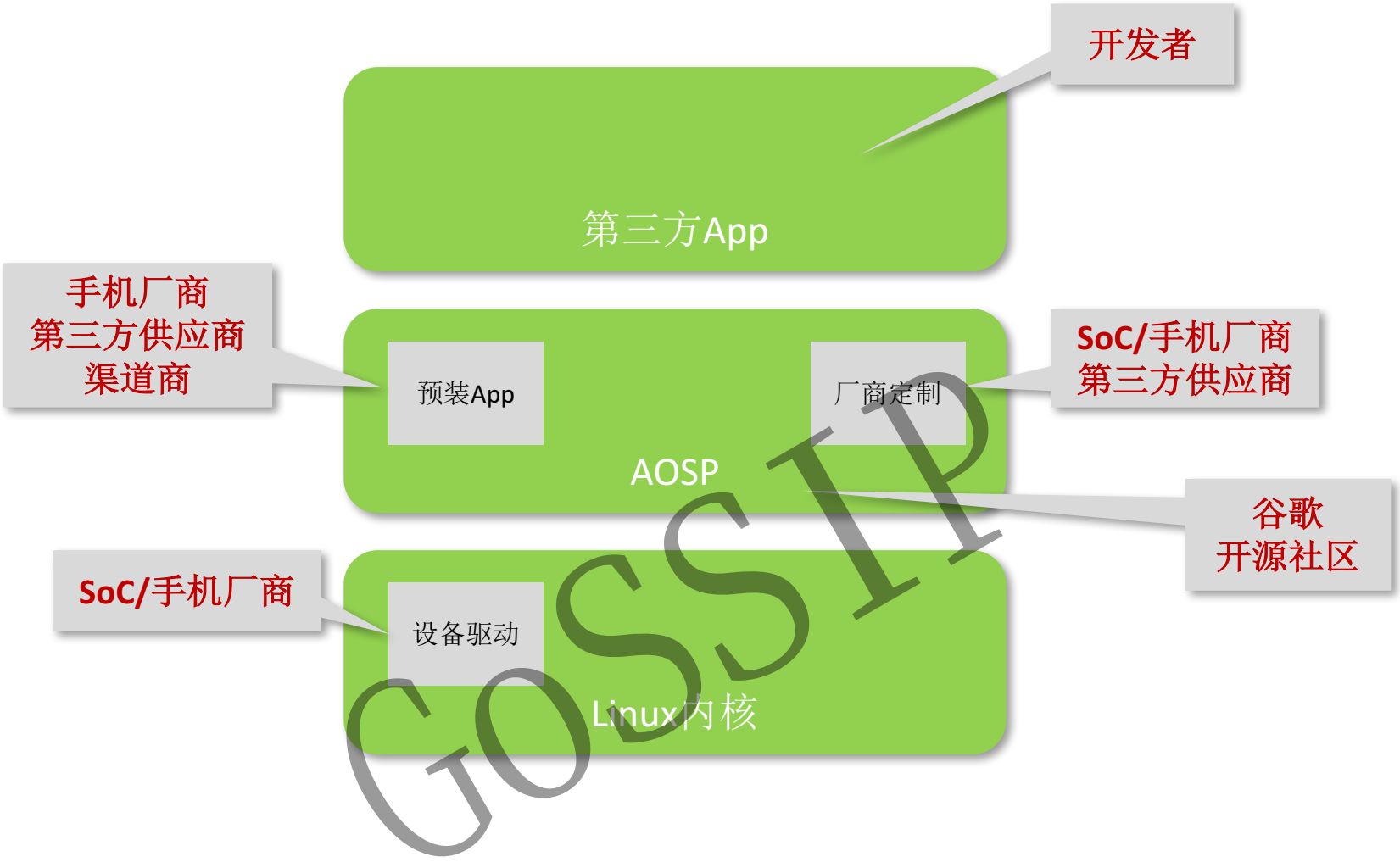
6 Replies

The Futex bug (CVE-2014-3153) is a serious bug that affects most Linux kernel version and was made popular by geohot in his towelroot exploit. You can read the original comex report at hackerone. Others have succesfully implemented this (this one for example), but no public exploit source code is available.

This post will describe in detail about what exactly is the futex bug, how to exploit the futex bug, and also explains how towelroot works and what the modstring in Towelroot v3 actually do. Following the footsteps of other security researchers, I will not give a full source code to the exploit. By giving enough details, I hope programmers can learn and appreciate the exploit created for this bug. By not releasing the source, I hope this should stop most script kiddies. There will be some small details that I will gloss over (about the priority list manipulation), so it will require some thinking and experimentation to implement the exploit.

One thing to note: I did some kernel programming, but never written a kernel exploit before, so this is my first time, I hope this is a good write up for a newbie exploit writer like me. Distributing the exploit source code will be useful only to handful of people, but writing about it will be useful to all programmers interested in this.

“大杂烩”


开发者
第三方App
手机厂商
第三方供应商
渠道商
预装App
厂商定制
SoC/手机厂商
第三方供应商
AOSP
谷歌
开源社区
SoC/手机厂商
设备驱动
Linux内核
GOSSIP

木桶理论

任何一个短板都有可能造成威胁

谁是短板的那一块？

短板

- Linux内核
- 芯片驱动
- AOSP
- 厂商定制/预装应用
- 第三方应用

Quick Quiz

安卓系统的security boundary是什么？

Permission是如何enforce的？

Linux内核/芯片驱动篇

Linux内核

In fact, all the boring normal bugs are way more important....Security people are often the black-and-white kind of people that I can't stand. I think the OpenBSD crowd is a bunch of [self-stimulating] monkeys....

rw kernel text even in year 2014

Linux内核

内核和用户程序交互的接口是什么？

Linux内核

- CVE-2013-2094: perf-event (整数溢出)
- CVE-2013-6282: get_user/put_user
- CVE-2014-0196: pty/tty(隐藏了5年+)
- CVE-2014-3153: futex (UAF)
- CVE-2015-3636: Ping Pong (UAF)
- 未公开的zero-day漏洞

CVE-2013-2094

- 一行代码引起的漏洞

```
static int perf_swevent_init(struct perf_event *event)
{
    int event_id = event->attr.config;
    ...
   static_key_slow_inc(&perf_swevent_enabled[event_id]);
   ...
}
```

如果event_id为负数呢？

不要信任来自用户态的任何数据！

CVE-2013-6282

- put_user (x, ptr)

  x： Value to copy to user space.

  ptr: Destination address, in user space.

如果ptr为内核态地址呢？

```
 ENTRY(__put_user_4)
+       check_uaccess r0, 4, r1, ip, __put_user_bad
 4: TUSER(str)  r2, [r0]
        mov     r0, #0
        mov     pc, lr
 ENDPROC(__put_user_4)
```

CVE-2015-3636


```
2301  [ 3354.778717] Unable to handle kernel paging request at virtual address 00200200
2302  [ 3354.778839] pgd = ea574000
2303  [ 3354.778900] [00200200] *pgd=00000000
2304  [ 3354.779052] Internal error: Oops: 805 [#1] PREEMPT SMP ARM
2305  [ 3354.779144] Modules linked in:
2306  [ 3354.779266] CPU: 1    Tainted: G        W     (3.4.0-Kali-g006dd6c #1)
2307  [ 3354.779357] PC is at ping_unhash+0x50/0xd4
2308  [ 3354.779479] LR is at _raw_write_lock_bh+0xc/0x8c
2309  [ 3354.779541] pc : [<c08b18bc>]    lr : [<c09f7d9c>]    psr: 20010013
2310  [ 3354.779541] sp : e99a5ee0  ip : c08a67ac  fp : 00000000
2311  [ 3354.779724] r10: 00000000  r9 : e99a4000  r8 : c000e928
2312  [ 3354.779846] r7 : 0000011b  r6 : 00000053  r5 : 00000000  r4 : eb3cd200
2313  [ 3354.779907] r3 : 00000003  r2 : 00200200  r1 : 00000000  r0 : c144ed98
2314  [ 3354.780029] Flags: nzCv  IRQs on  FIQs on  Mode SVC_32  ISA ARM  Segment user
2315  [ 3354.780120] Control: 10c5787d  Table: ab97406a  DAC: 00000015
```

在用户态映射200200地址使得kernel继续往下执行，造成UAF！

为什么是0x200200?

芯片驱动

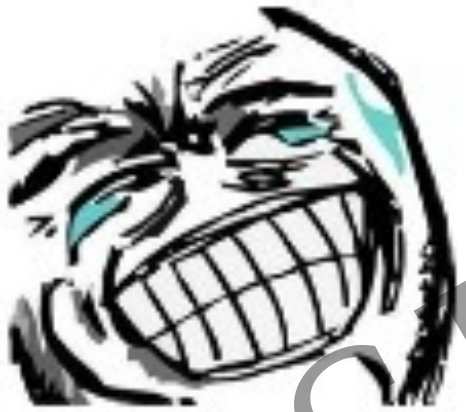
- ioctl：用户程序和内核交互的接口

不要信任来自用户态的任何数据！

```
1   int hifi_dsp_write_param(unsigned long arg)
2   {
3   ...
4       if (copy_from_user(&para, (void*)arg, sizeof(struct misc_io_sync_param))) { //arg用户可控，拷贝到para
5           ...
6       }
7       ...
8       // -> 分配堆内存，大小SIZE_PARAM_PRIV
9       hifi_param_vir_addr = (unsigned char*)ioremap(hifi_param_phy_addr, SIZE_PARAM_PRIV);
10      ...
11      // -> 未对para传人的大小进行校验，可能造成堆溢出
12      ret = copy_from_user(hifi_param_vir_addr, para.para_in, para.para_size_in);
13      ...
14  }
```

CVE-2012-6422

/dev/exynos-mem rw-rw-rw



新的进展

- 内核text变成只读：修改function pointer/data
- PXN: kernel ROP, addr_limit绕过
- SELinux
  - 漏洞存在，但是对驱动漏洞的利用变得困难
  - 需要多个漏洞联合使用

AOSP

面临的困难

- 代码来源复杂，完全控制比较困难
- 安全假设被破坏
- 开发人员的安全背景

一些基础知识

- UID based sandbox

app签名

- app在release之前需要被证书签名
  - 证书反映了一个开发者的身份过程对
- 具体过程
  - apk每一个文件计算SHA1，然后Base64编码 --- 生成 MANIFEST.MF
  - 对 MANIFEST.MF 签名(用私钥) --- 结果保存在 CERT.SF
  - 生成 CERT.RSA 文件，保存公钥，签名算法，证书所有人颁发人等等信息 (ASN.1 格式)

签名带来的问题

- MasterKey: Blackhat 2013
- FAKE ID: Blackhat 2014

MasterKey

- 安装时候校验apk和运行时使用apk代码不同
- 处理同名文件逻辑不同
  - 校验apk文件 – 代码A
  - 使用apk文件 – 代码B
- 如何利用
  - zip中创建两个classes.dex

FAKE ID

- 校验证书时候有逻辑漏洞
- AOSP中有依赖签名从而加载plugin的代码
- 两者结合: 任意代码执行

FAKE ID

- 证书链
  - 反映了证书之间的信任关系
  - A颁发证书给B: A给B背书,A是B的证书颁发者(issuer)
  - A用自己的私钥对B签名
  - 在验证B的时候: 找到它的Issuer，用issuer(A)的公钥和保存的签名来验证B的公钥的完整性

FAKE ID

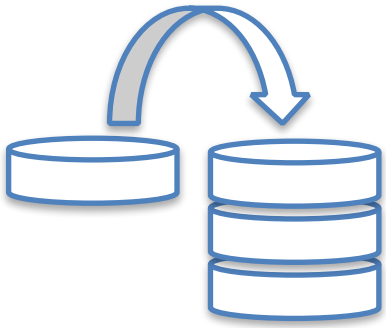
- Android没有用issuer的公钥验证B的公钥
- 恶意软件可以伪造issuer的公钥 (不是伪造B的公钥)以及其他的信息
- 伪造的issuer的公钥和其他信息会被保存在PackageInfo的signatures 中

FAKE ID

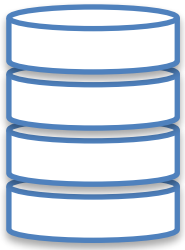
- 利用方法
  - 生成两个证书: A和malicious,A是malicious的issuer
  - 用malicious签名一个应用
  - 修改apk中的CERT.RSA中A的公钥:修改成adobe的
- 安装时:校验通过
  - apk是由malicious签名的,android不会使用issuer的公钥去校验malicious公钥
- 运行时
  - 系统会认为apk是由adobe签名过的

StageFright

X + 1 > x ?

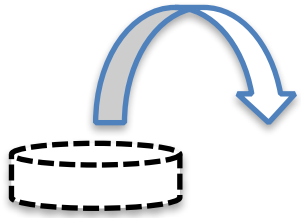


Allocating (Size+1)



I think I have (Size+1) :P

What you really allocated is (0xffffffff+1) = nothing!



What happens when you try to access it?

JDWPExposed

- 我们独立发现的漏洞: CVE-2015-3865:
- Android M 修复
- 原因: 又是一行代码引起的bug

```
293
294 // JDWP is allowed unless the Zygote forbids it.
295 static bool gJdwpAllowed = true;
296
```

- 安全假定： app一定是通过zygote起来

JDWPExposed

- 安全假定被打破
  - 应用可以通过app_process命令启动
  - 所有通过app_process启动的应用JDWP接口默认是打开的 (pm 命令)
  - 手机上特权应用通过app_process启动

```
root     1453  1     853652 24564 ffffffff 00000000 S lbesec.loader
```

新进展

- AOSP monthly security bulletin
- AOSP security bounty program

AOSP漏洞挖掘

- 代码审计
- 开源版本之间的比较
- 已知漏洞pattern的提取和匹配

厂商定制/预装app

厂商定制

- 为什么
  - 更好的feature
  - 本地化的需要
- 安全影响
  - 引入漏洞
  - 更慢的OTA周期

我们的研究

厂商	型号	发布日期	版本	厂商	型号	发布日期	版本
Google	Nexus S	12/2010	2.3.6	中兴	C N880	05/2011	2.2.2
	Nexus 4	11/2012	4.2		N881F	02/2013	4.0.4
三星	Galaxy S2	04/2011	2.3.4	华为	C8650+	09/2011	2.3.6
	Galaxy S3	05/2012	4.0.4		Ascend Mate	03/2013	4.1.2
HTC	Wildfire S	05/2011	2.3.5	酷派	8150	01/2012	2.3.7
	One X	05/2012	4.0.4		7295	03/2013	4.1.2
LG	Optimus P350	02/2011	2.2	联想	A65	02/2012	2.3.5
	Optimus P880	06/2012	4.0.3		K860	08/2012	4.2.1
索尼	Xperia Arc S	09/2011	2.3.4				
	Xperia SL	09/2012	4.0.4				

结果

漏洞类型	Google		三星		HTC		LG		索尼		中兴		华为		酷派		联想	
	Nexus S	Nexus 4	S2	S3	Wildfire S	One X	P350	P680	Xperia Arc S	Xperia SL	C N880	N881F	C8650+	Mate	8150	7295	A65	K860
后台电话	1	0	2	4	2	1	1	3	1	1	1	2	1	2	1	1	2	3
后台消息	3	2	7	7	6	3	4	4	3	3	5	2	5	2	5	2	3	4
静默安装	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	2	0	1
短信欺诈	2	0	4	4	5	6	3	3	2	2	2	2	3	1	4	5	5	4
清除数据	0	0	3	2	1	0	0	0	0	0	0	0	0	0	1	0	1	0
签名漏洞	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

短信欺诈

- 伪造任意短信
  - 安卓原生漏洞：4.2版本修复
  - 厂商定制引入

DEMO

S5 TransCloud


```
private void registerTranscloudInstallObserver() {
    Slog.v("Transcloud", "register transcloud install observer...");
    IntentFilter v0 = new IntentFilter();
    v0.addAction("android.intent.action.PACKAGE_ADDED");
    v0.addDataScheme("package");
    this.mContext.registerReceiver(new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            String v1 = intent.getData().getSchemeSpecificPart();
            String v0 = intent.getAction();
            if((v1.equals("com.samsung.android.service.transcloud")) && ("android.intent.action.PACKAGE_ADDED"
                    .equals(v0))) {
                Slog.d("Transcloud", "Transcloud installed!");
                TranscloudMonitorService.this.addTranscloudManagerService();
                TranscloudMonitorService.this.systemReadyTranscloudManagerService();
                TranscloudMonitorService.this.mContext.unregisterReceiver(((BroadcastReceiver)this));
            }
        }
    }, v0);
}
```

包名是可以被伪造的！

选择队友很重要

广升FOTA

了解更多FOTA功能

战略合作

联发科	展讯	ZTE中兴	华为	凡卓	华勤
希姆通	辉烨	天宇朗通	唯乐Vollo	奥克斯	夏新
卓普	朵唯	德赛	海尔	万利达	康佳
波导	海信	欧新	欧博信	爱立顺	TCL

选择队友很重要

- CVE-2014-1600

```
srw-rw-rw- system   system          2014-02-22 12:49 fotabinder
```

- Every normal app could connect to this socket to send commands to the service and execute commands as ***root***!
- google released a CTS test case to detect this problem

选择队友很重要

- CVE-2015-2231

srw-rw-rw- system system 2014-02-22 12:49 fota

- Create a socket and wati for commands from socket (command is encrypted with a hardcoded key – encryption != security)
- Normal app could connect to this socket and execute app as ***system(not root)***

选择队友很重要

- CVE-2015-2232

ls -l /dev/block/mmcblk0 brw-**rw**---- root **system** 179, 0 2015-03-09 13:41 mmcblk0

- System UID has full privilege to write to the whole paritition
- Change a script which will be executed as root at boot - pwned

选择队友很重要

- 第三方供应商app被预装在系统分区
  - 获取system级别的permission：静默安装
- WormHole漏洞
- 预装浏览器
  - 插件机制

定制/预装应用的漏洞挖掘

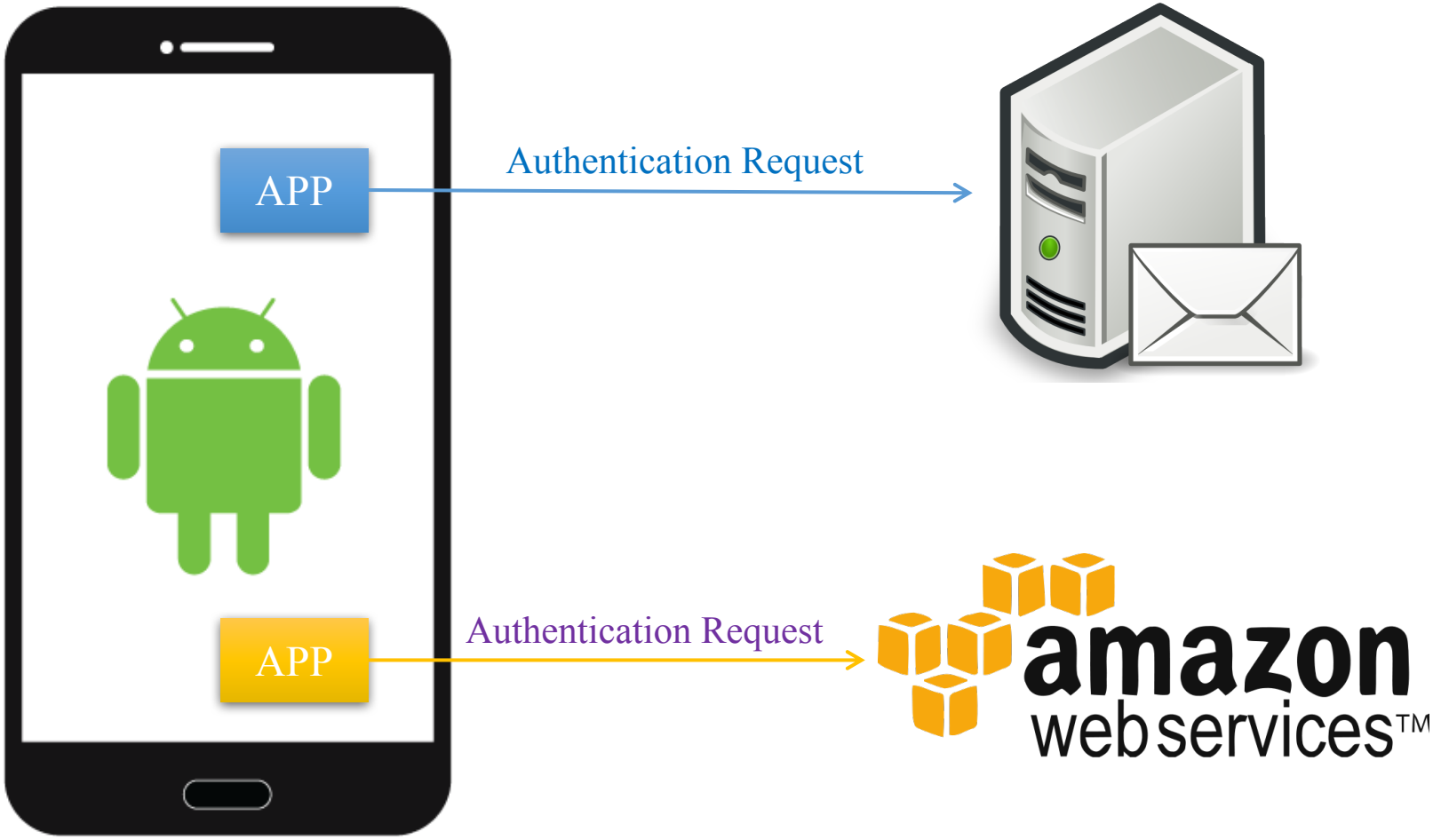
- 基本的检测工具
  - 文件属性
  - Socket端口
  - 组件接口
- 安全思维
  - 网络环境可信吗
  - 加密 != 安全
  - 业务逻辑的安全梳理

第三方app

第三方app

- 开发者缺少安全意识
  - Content provider的保护
  - Credential的保护

Third-party Services Require Authentication



Protecting Developer Credentials is Hard


```
1  .method public static SendMailInBackground
2   new-instance v3, Lcom/pompeiicity/funpic/Email;
3   const-string v7,  "whav*****@gmail.com"
4   const-string v8,  "jea****"
5   invoke-direct {v3, v7, v8}, Lcom/pompeiicity/funpic/Email;->
6                          <init>(Ljava/lang/String;Ljava/lang/String;)V
7   ...
8  .end method
```

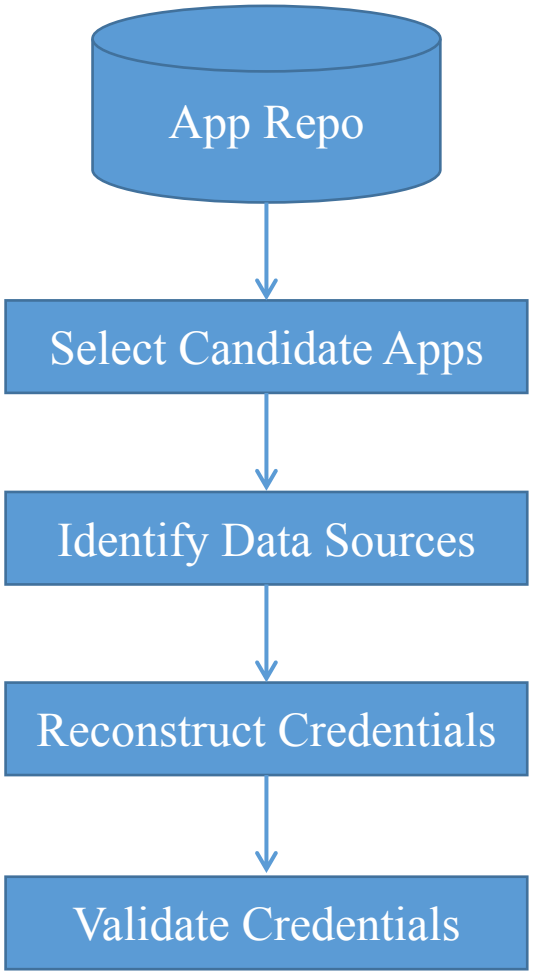
Credential Leak is Dangerous

"whav*****@gmail.com"

"jea****"

.....

CredMiner: Mine Credentials from Apps



Select Candidate Apps

- Apps that use interesting libraries (i.e., libraries that accept ***plaintext*** credentials)

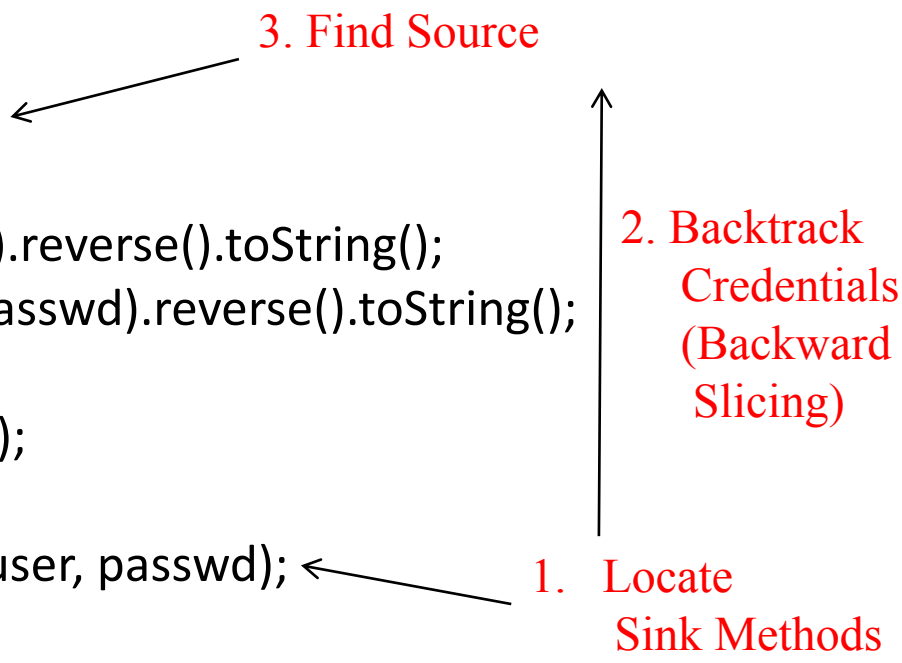
JavaMail Library

Amazon AWS Library

……

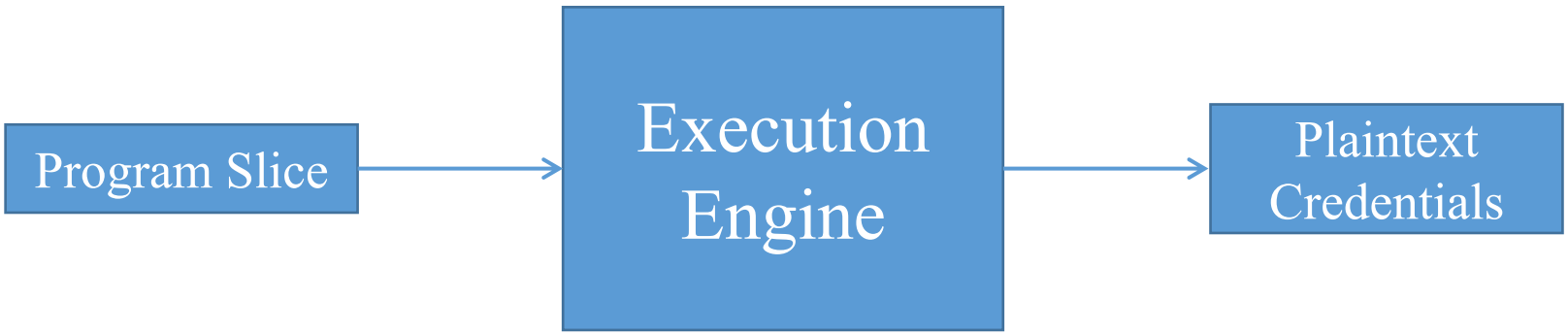
Identify Data Sources


```
String _user,  _passwd;

_user = “edcba"; _passwd = “54321";

String user = new StringBuilder(_user).reverse().toString();
String passwd = new StringBuilder(_passwd).reverse().toString();

System.out.println(“Authenticating…");

return new PasswordAuthentication(user, passwd);
```



Reconstruct Credentials

- Use an execution engine (in Python) to execute (forward) the program slice.
- Create mock objects on demand, to run the program slice.



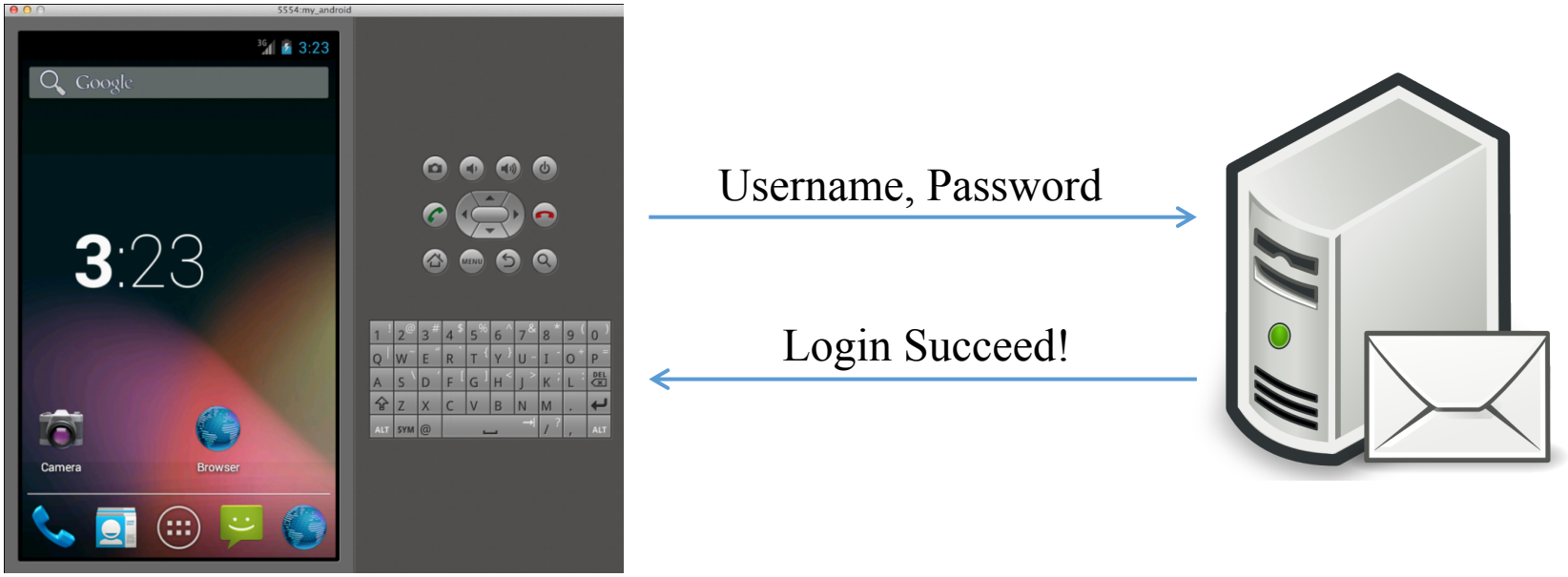
Reconstruct Credentials: smali vm

Emulated java.lang.StringBuilder.append(char):

```
1  def StringBuilder_append_C_Ljava_lang_StringBuilder(arg_regs):
2    #get the register value. Register names are in the
3    #parameter arg_regs, v1, v2 for instance
4    arg1_value = vm.get_reg_value(arg_regs.split(",")[0].strip())
5    arg2_value = vm.get_reg_value(arg_regs.split(",")[1].strip())
6    #check the type
7    if type(arg1_value) != str or type(arg2_value) != str:
8      print ("[x] [%d] we expect an (string, string) here. But"
9             "the actual type is (%s,%s)" % (get_linenumber(),
10            str(type(arg1_value)), str(type(arg2_value))))
11     return False
12   #emulate java.lang.StringBuilder.append(char)
13   arg1_value += arg2_value
14   vm.put_reg_value(arg1, arg1_value)
15   return True
```

Validate Credentials

- Run the app in an Android emulator and monitor its execution.
- Compare the run-time parameters to those recovered by CredMiner.
- Monitor the interaction with remote servers.



Evaluation

Distribution of Collected Apps:

	Google Play	Alternative Markets
# of Apps	21, 092	15, 479
Percentage	57.67%	42.33%
# of Total Apps	36, 571	
# of Distinct Apps	36, 561	

Evaluation

Overall Result:

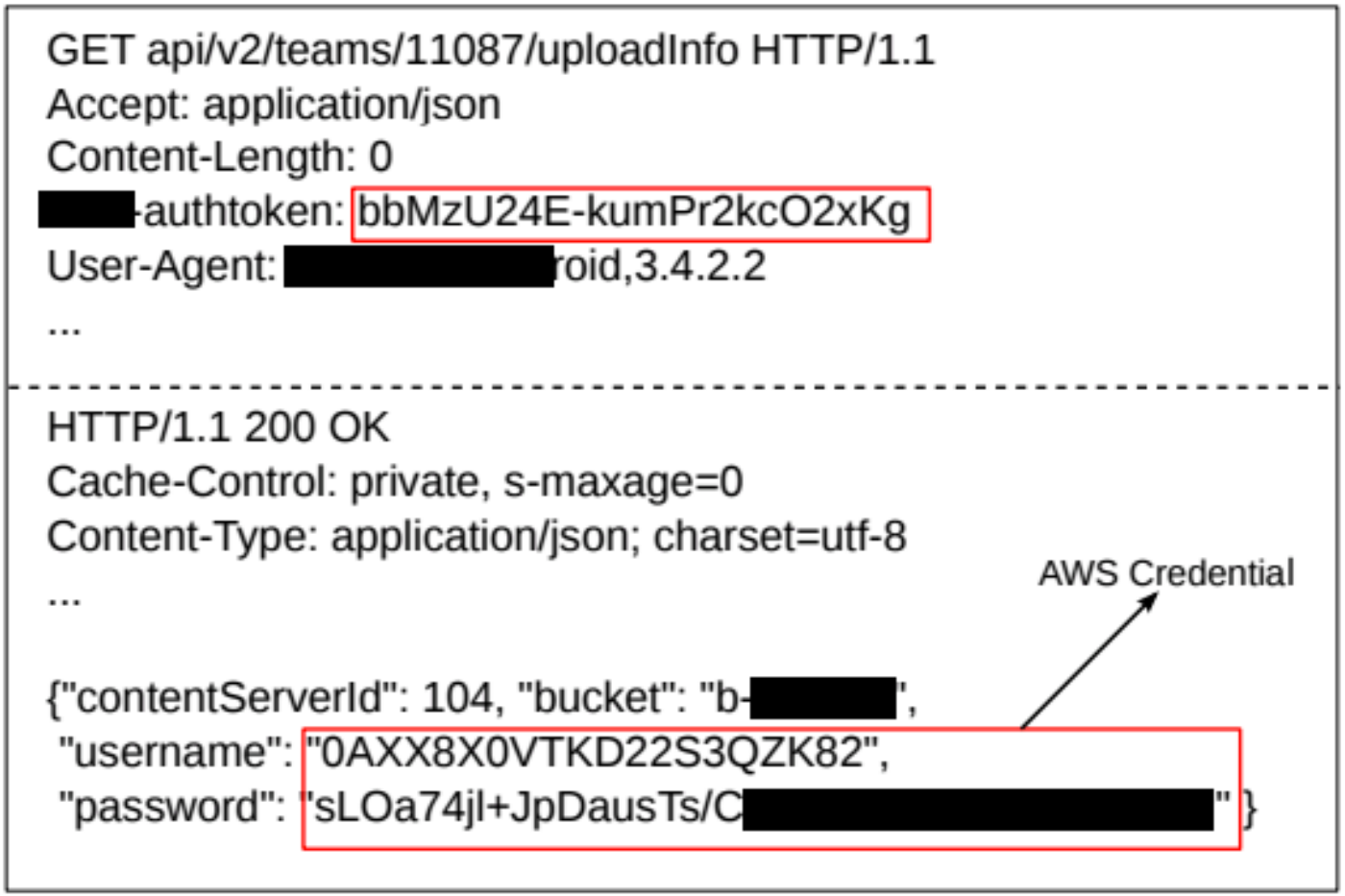
- 237 candidate apps use the JavaMail library.
- 196 candidate apps use the Amazon AWS SDK.
- 51.1% (121/237) and 67.3% (132/196) of these candidate apps are vulnerable.
- Distribution of Vulnerable Apps:

Category of Credentials	Google Play	Other Stores	Total
Email Credentials	65	56	121
Amazon AWS Credentials	128	4	132

Evaluation


```
public DefaultMail(String toMail, String subject, String body) {
        this.emailAccount = new String[]{
                "iengnathgran@126.com", "tsiudaschn@126.com", "yvababstvenu@126.com",
                "phtreddrick@126.com", "cqcancinor@126.com", "xpmonohanschwa@126.com",
                "tfkleinb@126.com","vcolganbird@126.com", "dsimentals@126.com",
                "jnconole@126.com", "tdlneebarria@126.com","pbmammenj@126.com",
                ...
        };
        this.emailPassword = new String[]{
                "qpvr[redacted]", "hpf[redacted]", "cflv[redacted]",
                "fphv[redacted]", "tcj[redacted]","vkns[redacted]",
                "qinxb[redacted]", "mopr[redacted]", "fbfr[redacted]",
                "ynue[redacted]", "wk[redacted]", "ynx[redacted]",
                 ...
         };
}
```

Evaluation


GET api/v2/teams/11087/uploadInfo HTTP/1.1
Accept: application/json
Content-Length: 0
[redacted]-authtoken: bbMzU24E-kumPr2kcO2xKg
User-Agent: [redacted]roid,3.4.2.2
...
HTTP/1.1 200 OK
Cache-Control: private, s-maxage=0
Content-Type: application/json; charset=utf-8
...
AWS Credential
{"contentServerId": 104, "bucket": "b-[redacted]",
"username": "0AXX8X0VTKD22S3QZK82",
"password": "sLOa74jl+JpDausTs/C[redacted]" }

summary

- Never embed developers' credentials in the app.
- Use secure solutions from service providers correctly

新的趋势

移动设备新的应用场景

- 移动支付
- 指纹
- IoT
- 汽车

Q&A