

Resurrecting The READ_LOGS Permission On Samsung Devices

Ryan Johnson, Kryptowire / GMU

Angelos Stavrou, Kryptowire

Agenda

- android.permission.READ_LOGS permission
- Sensitive data written to the Android log
- How to regain the READ_LOGS permission on Samsung devices
- Why certain Samsung builds write notifications to the Android log
- Threat mitigation

Android Log

- A shared resource that any process can use
- Google apps, user apps, and Android OS processes can write sensitive data to the log
- android.util.[Log, Slog, EventLog]
- Generally read with logcat
- Full access requires user belong to the log group

android.permission.READ_LOGS

- READ_LOGS allows access to various device logs
- Android 4.1 and up, this permission stopped being granted to user apps (non-system apps)

```
<!-- @SystemApi Allows an application to read the low-level system log files.  
<p>Not for use by third-party applications, because  
Log entries can contain the user's private information. -->  
<permission android:name="android.permission.READ_LOGS"  
    android:permissionGroup="android.permission-group.DEVELOPMENT_TOOLS"  
    android:protectionLevel="signature|system|development"  
    android:label="@string/permlab_readLogs"  
    android:description="@string/permdesc_readLogs" />
```

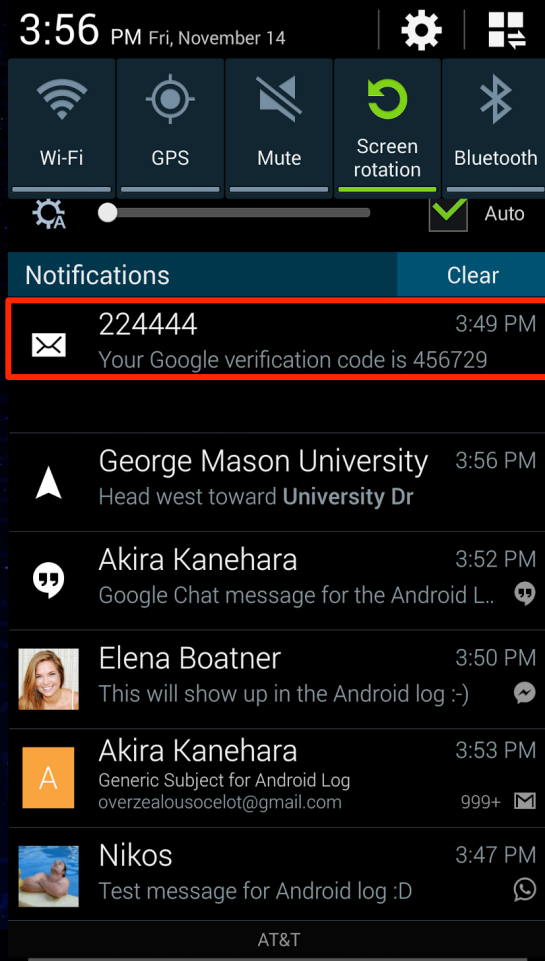
Why regain READ_LOGS perm?

- User's email addresses
- Cell-tower ID (which can get an approximation of their location using <http://opencellid.org/>)
- Raw GPS data
- Non-Google account names (e.g., Twitter)
- Cellular network and Wi-Fi information
- Voicemail number
- The name associated with a Gmail account
- Google IDs (which can be used to view Google+ page)
- Being able to tell whether the user is present or not

Why regain READ_LOGS perm?

- Integrated Circuit Card Identifier (ICCID)
- International Mobile Subscriber Identity (IMSI)
- Mobile Country Code Mobile Network Code (MCCMNC)
- Device serial number
- MAC address
- URLs and GPS coordinates opened via an Intent with the `android.intent.action.VIEW` action string
- URLs in Chrome for which there are errors
- Which apps the user or system executes and when

Notifications in the Android Log



```
11-14 15:49:07.983 813 1163
I notification_enqueue:
[com.android.mms, 123, NULL,
0, Notification(pri=2
icon=7f0202c3
contentView=com.android.mms /
0x1090086 vibrate=null
sound=content://settings/
system/notification_sound
defaults=0x4 flags=0x1
when=1415998146000 ledARGB=0x0
contentIntent=Y deleteIntent=Y
contentType=224444
contentText=Your Google
verification code is 456729
tickerText=224444: ,\u00c2\u201cYour
Google verification code is
456729 kind=[android.message]) ]
```

Write all the sensitive data to the log

- A certain Fitness & Health app on Google Play writes the following data to the Android log
 - username, password, cookies, insurance information, your medical procedures, your medications, and your conditions
- 5,000,000 – 10,000,000 installations
- Around 85,000 reviews
- Informed the company

Write all the sensitive data to the log

User Login

MARK : appboy login stuff (my ***** login): 1237397, **feelsgoodman@gmail.com**

User Password

JOSH : key 1 generated from password: ***trap_Door[190***

Cookie

j : **Cookie:**

*******_session_tracker=f1ebcf48939fb7968ebe962ff6c46b55.1414426817;**
*******_unique_tracker=75a4db35d1bd977ddd3cff3acfe9fc3d.1414426817;**
*******_session=8aa8c6bf4d5be5f2c83ab523e77130355c4fc3f575da31fef08d8a458f**
5d4e4bbd5e93c9d65c1463a3cb59541df5550e02148f5ad2f2256094b2f89412f6d0fb;

env=production

j : **Cookie2: \$Version=1**

j : **X-CSRF-Token: YmawWVNBGFFvojUTev65opDRme7UB20KFq3r2iScjAs=**

Write all the sensitive data to the log

Security Token

REQUEST PAYLOAD : https://healthnews.*****health.com/preferred_articles.json?installation_id=e6608bc8-f652-425c-b02e-512e59b665fb&security_token=704e16fc5d724657e34d1665c83ac39a2da5c121&page=1&per_page=25&last_updated_since=1414165392.879

Medical Procedures

MARK : {"id":340,"item_note":"**Finally**","procedure_date":"10/27/2014","_etag":"","procedure_doctor":"Dr. Robot","_deleted":false,"name":"**Buttocks lift**"}

Medications

DUCK : {"4d754d1b-dbbe-4d04-8df9-61df080e130f":{"id":609,"medication_doctor":"Dr. Robot","name":"**Diazepam**","item_note":"","medication_date":"10/27/2014","_awaitingUpload":true,"_deleted":false,"dosages":[{"dose_type_id":3,"amount":"**100mg**","method_type_id":5,"frequency":"**All the time**"}]}}

Write everything to the Log

Conditions

```
DUCK : {"0797760b-433b-494d-b5d3-a5b71909b931":{"id":  
10,"item_note":"O_o","_awaitingUpload":true,"_deleted":false,"dis  
ease_date":"10\27\2014","disease_doctor":"Dr.  
Robot","name":"Acid (LSD) abuse"}}
```

Insurance Information (Aetna Group Number W3342212105

Member ID: 1234567-123-12345)

```
z : https://www.*****health.com/api/v1/narrow_network/33/  
validate_member_info/W3342212105/1234567-231-12345 ->
```

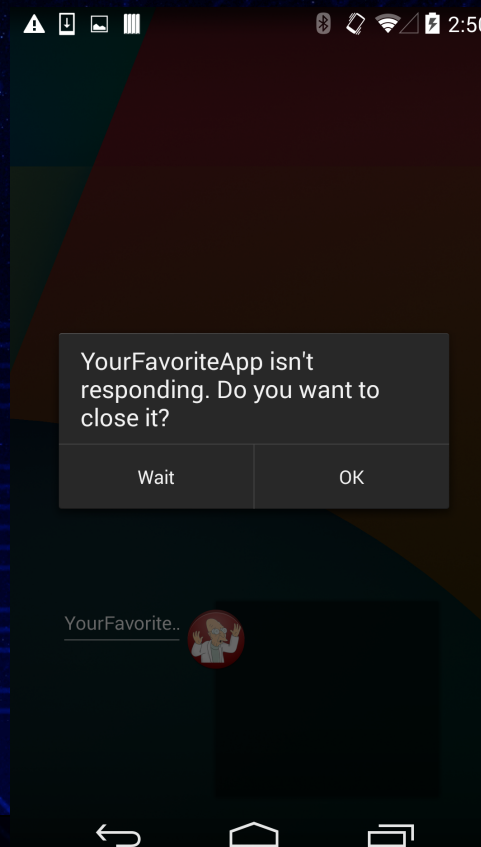
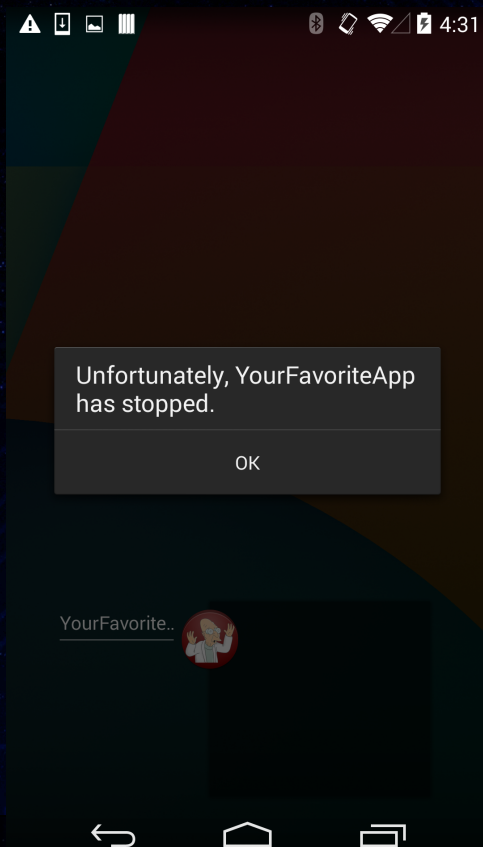
HTTP/1.1 401 Unauthorized

Regaining Android Log Access

- A world-readable log file is generated when any of the three conditions occur on Samsung devices
 - Uncaught exception in app's Dalvik bytecode
 - Application Not Responding (ANR) event
 - Error encountered in an app's native code library
- The first two will generate a system message and the third will not if done properly
- Only appears to work on Samsung devices

Three Different Bricks

- Uncaught exception, ANR, Native code error



dumpstate binary

- dumpstate binary collects and writes system-level data when something goes wrong
- Cannot be run by user apps
- Contains Android log, kernel log, system properties, data from the proc file system, etc.
- Generally creates about 3 to 6 MBs of output
- `/system/bin/dumpstate`

Error in native code

- Java Native Interface (JNI) to call C function
- Fork the process in the C function
 - Call abort() in child process
 - Return in parent process
- Sends SIGABRT signal which is handled by debugerd and calls dumpstate
- dumpstate will gather system information and write to `/data/log/dumpstate_app_native.txt.gz`
- Does not generate any error message

Dumpstate Files on Samsung Android

- Dumpstate files written to /data/log

```
-rw-r--r-- shell log 1007511 2014-10-12 15:51 dumpstate_app_anr.txt.gz  
-rw-r--r-- shell log 605572 2014-10-18 20:54 dumpstate_app_error.txt.gz  
-rw-r--r-- shell log 568151 2014-10-20 23:37 dumpstate_app_native.txt.gz
```

- The dumpstate files are readable by any user on the device
- `dumpstate -k -t -z -d -o /data/log/
dumpstate_app_native -m 8028`

/data/log directory

- From init.rc file from the KOT49H.I9500XXUGNJ1 build (Android 4.4.2)

```
# SA, System SW, SAMSUNG create log directory
```

```
mkdir /data/log 0775 system log
```

```
chown system log /data/log
```

```
mkdir /data/anr 0775 system system
```

```
chown system system /data/anr
```

```
chmod 0775 /data/log
```

```
chmod 0775 /data/anr
```

```
restorecon /data/log
```

```
restorecon /data/anr
```

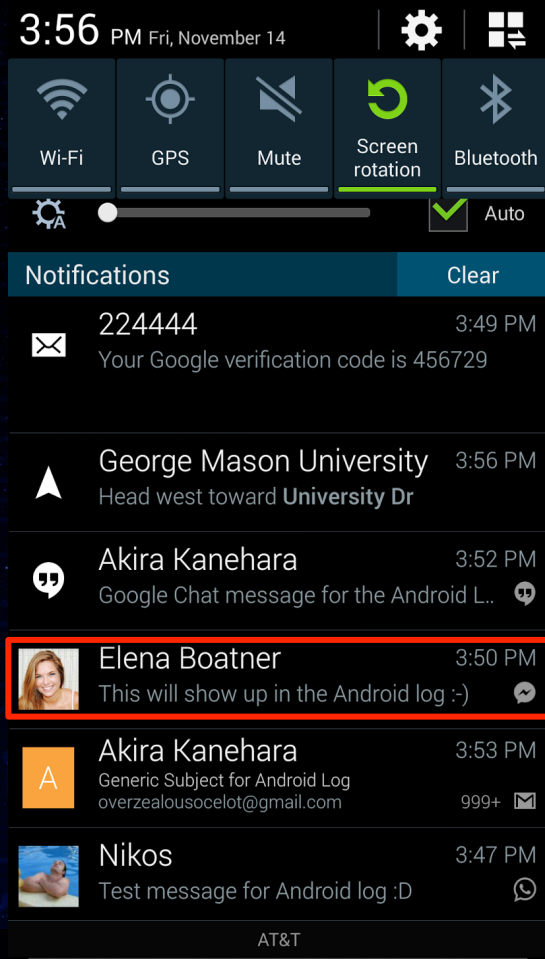
dumpstate file snippet

```
[0,14585,10012,com.android.contacts,broadcast,com.android.contacts/  
com.sec.android.app.contacts.ContactsReceiver]  
11-14 15:47:29.527 813 1435 I am_create_service:  
[0,1170423648,.GoogleAccountDataService,10061,1597]  
11-14 15:47:29.547 813 824 I notification_enqueue: [com.whatsapp,1,NULL,  
0,Notification(pri=0 icon=7f0205c2 contentView=com.whatsapp/0x1090086  
vibrate=null sound=null defaults=0x0 flags=0x0 when=1415998049288 ledARGB=0x0  
contentIntent=Y deleteIntent=N contentTitle=Nikos contentText=Test message for  
Android log :D tickerText=Message from Nikos kind=[null])]  
11-14 15:47:29.567 813 1392 I am_proc_bound: [0,14585,com.android.contacts]  
11-14 15:47:29.617 813 1163 I am_proc_start:  
[0,14600,10015,com.sec.android.provider.badge,content  
provider,com.sec.android.provider.badge/.BadgeProvider]  
11-14 15:47:29.657 813 1472 I am_destroy_service: [0,1170423648,1597]  
11-14 15:47:29.657 813 1406 I am_destroy_service: [0,1171596560,1597]  
11-14 15:47:29.657 813 1461 I am_destroy_service: [0,1173214336,1597]  
11-14 15:47:29.667 813 813 I notification_cancel: [android,2,NULL,0,0,0]  
11-14 15:47:29.667 813 1438 I am_proc_bound:  
[0,14600,com.sec.android.provider.badge]  
11-14 15:47:29.738 813 22582 I power_partial_wake_state: [0,ActivityManager-  
Launch]
```

Create exploit application

- Create app with persistent execution
 - `android.permission.RECEIVE_BOOT_COMPLETED`
- Start Service application component at boot
- Use JNI to call C code that causes an error in native code at scheduled intervals
- Uncompress and process the dumpstate file
- Exfiltrate
- Only evidence is are some log messages

Notifications in the Android Log



```
11-14 15:50:02.517 813 1391
I notification_enqueue:
[com.facebook.orca,10000,t_mid.
1415995249174:9405c32ed74e69fa7
9,0,Notification(pri=1
icon=7f0207a9
contentView=com.facebook.orca/
0x1090086 vibrate=null
sound=null defaults=0x0
flags=0x1 when=1415998201262
ledARGB=0xff00ff00
contentIntent=Y deleteIntent=N
contentTitle=Elena Boatner
contentText=This will show up
in the Android log :-)
tickerText=Elena Boatner: This
will show up in the Android
log :-) kind=[null] 1 action)]
```

android.app.Notification.toString()

- Samsung's android.app.Notification.toString method creates a much more verbose output than the corresponding Android Open Source Project (AOSP) method
 - Samsung's android.app.Notification.toString method includes the following instance variables in the string representation of the Notification object
 - `contentTitle`, `contentText`, and `tickerText`

Samsung vs AOSP Notifications

- AOSP
 - Notification(pri=0 contentView=com.kryptowire.bha/0x1090064 vibrate=null sound=null defaults=0x0 flags=0x0 kind=[null])
- Samsung
 - Notification(pri=0 icon=7f020000 contentView=com.kryptowire.bha/0x1090086 vibrate=null sound=null defaults=0x0 flags=0x0 when=1415746428600 ledARGB=0x0 contentIntent=N deleteIntent=N **contentType=Generic Title contentType=Generic Subject tickerText=Here is a Message kind=[null])**

NotificationManagerServer

- `com.android.server.NotificationManagerService.enqueueNotificationInternal` method snippet

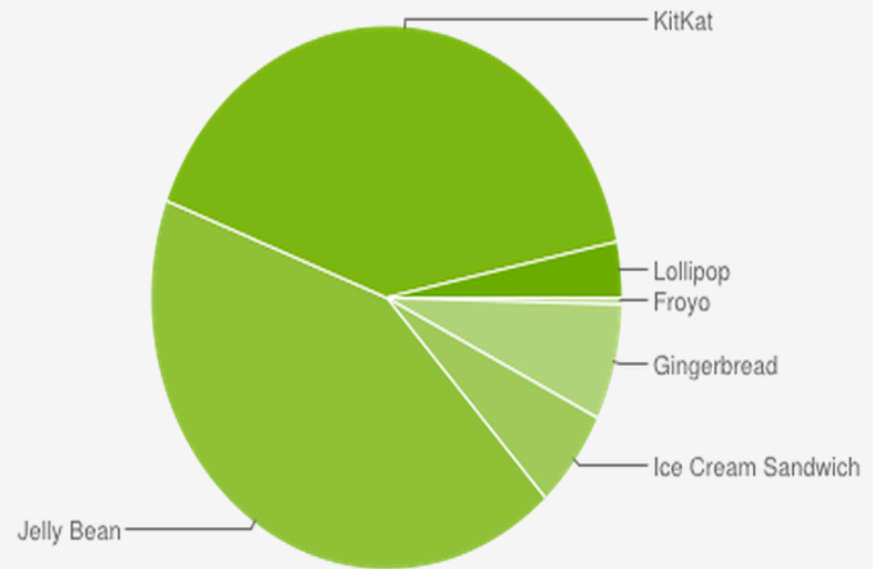
```
// This conditional is a dirty hack to limit the logging done on
// behalf of the download manager without affecting other apps.
if (!pkg.equals("com.android.providers.downloads")
    || Log.isLoggable("DownloadManager", Log.VERBOSE))
{
    EventLog.writeEvent(EventLogTags.NOTIFICATION_ENQUEUE,
    pkg, id, tag, userId, notification.toString());
}
```

Vulnerable Builds

- Downloaded firmwares from <http://www.sammobile.com/firmwares/>
- Flash tar.md5 file to device using Odin
- The notification vulnerability was found in certain Samsung Android 4.1.2, 4.3, and 4.4.2 builds, but it was fixed in the Android 4.4.4 FOTA update

Android Platform Usage

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x		17	19.4%
4.3		18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%



Data collected during a 7-day period ending on March 2, 2015.
Any versions with less than 0.1% distribution are not shown.

Source: <https://developer.android.com/about/dashboards/index.html>

Threat Mitigation

- Use Android Debug Bridge (ADB) to change the file permissions of the dumpstate files
- Owner is shell and group is log for these files
 - Same as ADB
- `touch dumpstate_app_native.txt.gz.tmp`
- `chmod 000 dumpstate_app_native.txt.gz.tmp`
- dumpstate changes from the root user to the shell user
- It uses output redirection to overwrite the file, but it will not be able to write to it due to the file permissions

Conclusion

- Sanitize sensitive data before writing it to the Android log
- Be careful when extending AOSP code
 - Test and audit carefully
- Be mindful of file permissions on files that contain sensitive data

Questions and Discussion

Thank you!

Ryan Johnson

rjohnson@kryptowire.com

<http://www.kryptowire.com/>