bindump

The bindump utility is a little tool I've put together for my book, after realizing that there's no tool I know of, AOSP or other, which can map out which PIDs communicate over Binder with which. I needed this type of mapping for the chapter dealing with the Framework Services, as well as the long discussion on Binder Internals. I'm still waiting on a few L changes before I can release the book (believe you me, it's been a long while in the making), but the tool is usable on its own. And is super simple, yes, though darn useful.

You can get the tool right here. Just unpack the tar. The binary is position independent, so it runs fine on all version of Android I've tested as of ICS up to and including L. As usual, you might want to check out the RSS feed, or follow my company's Twitter for more updates.

Why should I care?

If you've called Context.getSystemService(..), you've been using Binder (through proxy objects). The owning PID of most of these services is system_server, but not always - notable exceptions being the phone and batterypropregistrar. Vendor installed services sometimes also run in their own processes.

If you want to get a better idea of who's talking to whom in your device - this is a pretty good tool for that. From the /proc filesystem perspective, all you'll see is a file descriptor to /dev/binder - but this tool goes a step further to show you the actual Binder sessions. At this point, only for published services. I'll work on updating that by the time the book is out.

So how do I use it?

Glad you asked. Usage couldn't be simpler. Note you do not need any privileges and/or rooting for this to work!

```
shell@htc_m8wl:/ $ /data/local/tmp/bindump
Usage: /data/local/tmp/bindump [owner|users] _servicename_
Can also use "all" for all services
```

So you just put it a service name, for example:

```
shell@htc_m8wl:/ $ /data/local/tmp/bindump phone
Service: phone node ref: 785272
User:  PID 17355       com.htc.widget.process2
User:  PID 17158       com.google.android.music:main
```

```
User:  PID 31664      com.amazon.mp3
User:  PID 29473      com.vcast.mediamanager
User:  PID 29312      com.telecomsys.directedsms.android.SCG
User:  PID 27943      com.verizon.messaging.vzmsgs
User:  PID 26526      com.google.process.location
User:  PID 26306      com.htc.htcdialer
Owner: PID 26275        com.android.phone               # No surprise here :-)
User:  PID   366      /system/bin/servicemanager
```

you can filter by owner (only one per service) or users (1+ per service).

How does it work?

Ah. An even better question. While a discussion of Binder is outside the scope of this little write-up (and, hey, it's in the book!), this tool's functionality is simple enough to explain in a few lines:

- Bind to the servicemanager, and check for the requested service. This automatically establishes a binder connection to the service
- Next, seek out /sys/kernel/debug/binder/proc/$mypid. In it is a specific line of interest, in the form:

  ref xxxxxxx: desc yy node xxxxxxx s 1 w 1 d (null)

  That "node" is what we're looking for. There are actually two node - ones for the service manager, which we can skip - it's always the first one. The second one is the one binding to our service. So we can extract the node number easily.
- Finally, enumerate all entries in /sys/kernel/debug/binder/proc - these are all processes which have binder handles. If they have the node as a ref, that means they're users of this very same service. If they have the Node in a Node: line, they are the owner.

Example: healthd (owner of batterypropreg):

```
shell@htc_m8wl:/ $ cat /sys/kernel/debug/binder/proc/364
binder proc state:
proc 364
  thread 364: l 02
  node 17: u018317a0 c01830704 hs 1 hw 1 ls 0 lw 0 is 2 iw 2 proc 25871 366
  ref 15: desc 0 node 1 s 1 w 1 d  (null)        # Ref to ServiceManager
  ref 777219: desc 1 node 777218 s 1 w 1 d ed9b5600 # Owner of batterypropreg
```

And that's that. The full source is in the tar file (I'm not a fan of git in general nor github in particular). For the lazy, you can just click here. The source is nothing fancy - it takes that of service.cpp as a point of departure, but then embarks on its own journey. And yes, it uses strstr() for searching inside the binder data. So what? It's a PoC.

I'm working on improvements to adapt this to non published services (i.e. any binder

nodes), as well as produce SVG output. And I'm always looking for suggestions. So stay tuned.

When will it not work?

- If you don't have the debugfs mounted under /sys/kernel/debug
- If Binder wasn't compiled with debug data (which it is, by default)

Shameless plug for the book, and RFC

The book has a TON of detail about the framework services, and volume II covers the nooks and crannies of Binder in places where I betcha Dianne Hackborn herself (Love your work!) might feel uncomfortable ;-). The book is not out yet (but soon, really - I just need L to stabilize..). But in the interim, I encourage you to try out the tool (as well as the even more powerful Dexter) and shoot me an email (to j@) if you've any questions. Or comments. Or in general. All are welcome.