Binary Blobs Attack!!!

Zach Pfeffer (pfefferz)
Linaro Android Platform Team

Embedded Linux Conference 2012

I'm working on a movie...

Here's the script...

Working Title: Binary Blobs Attack!!!

8 AM - San Jose

Philip

...stretches, pees, starts checking mail

9 AM

Philip

"okay, I need to write a message through this blob to the core"

11 AM

Philip

"Crap! What's wrong, why did you crash?"

1 PM San Jose
2:30 AM Bangalore

Philip

"Yeah, I'm not sure, I send the message and BOOM! The core just stops."

FAE of FaST

"I think that teams in India, I'll send a mail"

Philip

"Okay..."

10 AM Bangalore, the next day
8:30 PM San Jose
Vishal
"I got your mail..."
FAE of FaST
NDA mumble, NDA mumble, NDA mumble...
Vishal
"right, you need to send the version of the interface you want before using it"
FAE of FaST
NDA mumble, NDA mumble, NDA mumble...
Vishal
"no you have the v2.2 of the doc, its on page 34x-rwfv-444-:) of version 2.2.0.1, check Dora for the new version"

11 PM San Jose
FAE of FaST
opens beer
"let me send this before I go to bed..."

8 AM San Jose
9:30 PM Bangalore
Philip
"okay, I sent the version"
runs the unit...
"crap! what's wrong with the graphics stack"
calls Ngraphics' FAE...

A Word Problem

Father's Day is in 13 days. SuperCoolNotGoogle and FaST is working on the FathersDayChecksYourFridgeHasBeer phone. It estimates that it will need to solve 325 crashes. 2 engineers and 1 FAE can solve 1 crash per day. How many Engineers and FAEs does SuperCoolNotGoogle need to ship their product?

325 crashes until launch

13 days until launch

325 crashes / 13 days = 25 crashes / day

2 engineers + 1 FAE = 1 crash / day

Answer

50 engineers + 25 FAEs

My movie wasn't picked up :(

Maybe I'll do a web series...

Here's the web series script...

Working Title: Binary Blobs Attack Again!

8 AM - San Jose

Philip

...stretches, pees, starts checking mail

9 AM

Philip

"okay, I need to write a message through this blob to the core"

11 AM

Philip

"Crap! What's wrong, why did you crash?"

1 PM San Jose

Philip

looks at source, puts a print in at the remote call, checks log
"Hmm... I guess I need to send the version first"
work, work, work
"Bingo!"

Another Word Problem

Mother's Day is in 13 days. SuperCoolNotGoogle is working on the MothersDayRemindsYourKidsToCall phone. It estimates that it will need to solve 325 crashes. SuperCoolNotGoogle has the source to everything so 1 engineer can solve 2 crashes per day. How many Engineers does SuperCoolNotGoogle need to ship their product?

325 crashes until launch

13 days until launch

325 crashes / 13 days = 25 crashes / day

2 engineers = 2 crashes / day

Answer

12.5 engineers

Wow!!!

Having the source allowed SuperCoolNotGoogle and FaST to save 12.5 engineers and 25 FAEs.

Maybe they could get 2 phones out instead of 1 or...

Time spent on binary blob production, integration and debugging has eclipsed any other activity in the product cycle.

Engineer Rants

- “Binary blob scribbling is impossible to debug”
- “Kernel upgrades mean a recompile cycle with the SoC's 3rd party vendor”
- “Binary blobs are un-ethical”
- “OS independent code is always buggy, slow and will only be used on the OS its not meant for”
- “I need debug symbols!”
- “Documentation, Helloooo”
- “I'm going to kick ARM out of the kernel”
- “I can't track Linus' tip because I'll lose enablement”
- “I need a JTAG”
- "Write once, run everywhere doesn't really work for Java, so how well is that going to work for a driver?"

Problems

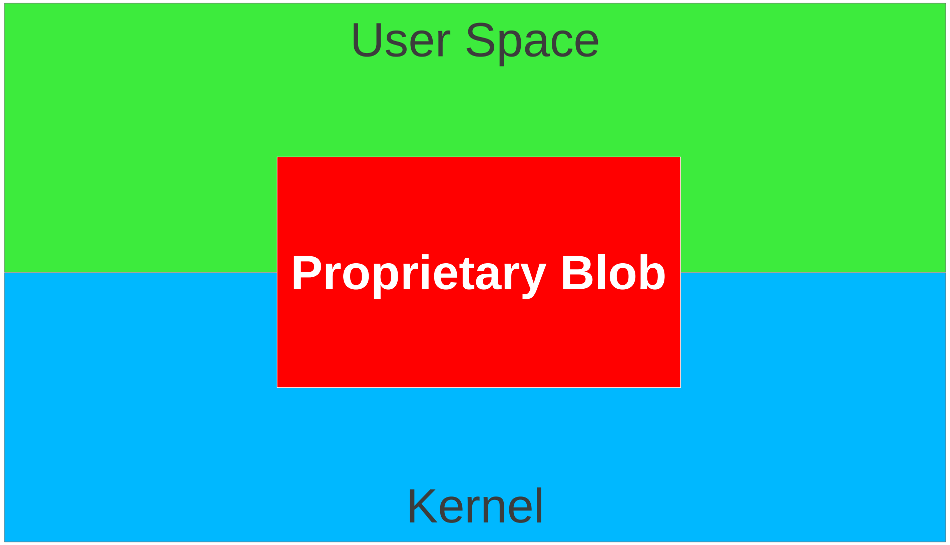
- The audio blob is locked to the 3.0 kernel the graphics blob is locked to 3.1
- hardfp and other toolchain improvements are lost
- The blob producers are impossible to track down
- Has brought back “batch” development

Binary blobs are responsible for the Linux ARM “problem.”

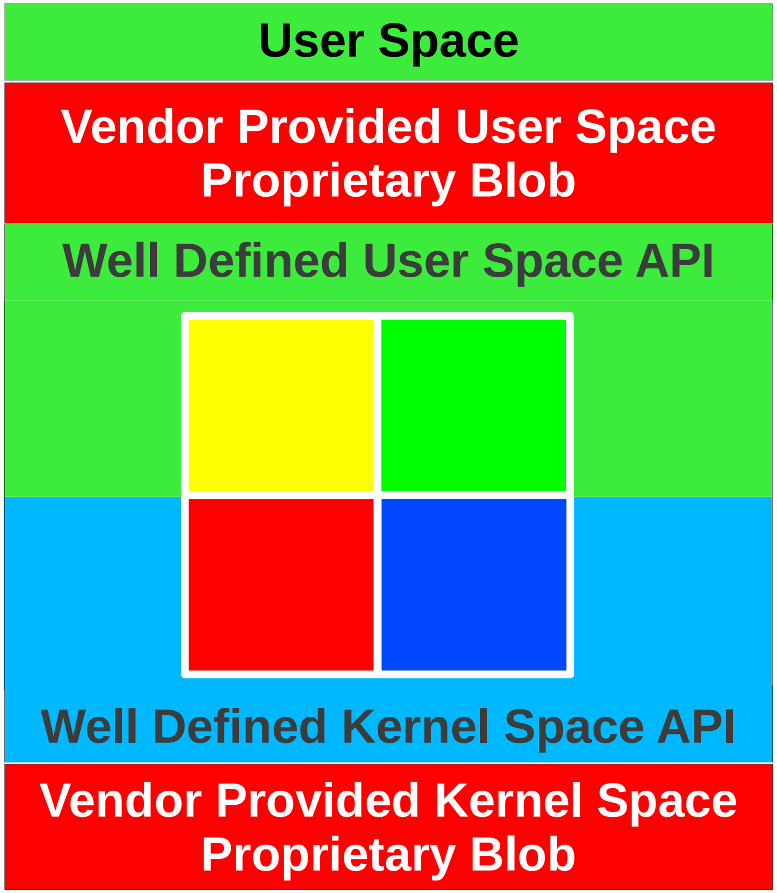
Binary blobs won't be going away.

How can we ensure an open, sustainable software eco-system with them?

Most problems happen when proprietary code crosses the userspace/kernel boundary



How Windows deals with this (so I've heard)



User Space

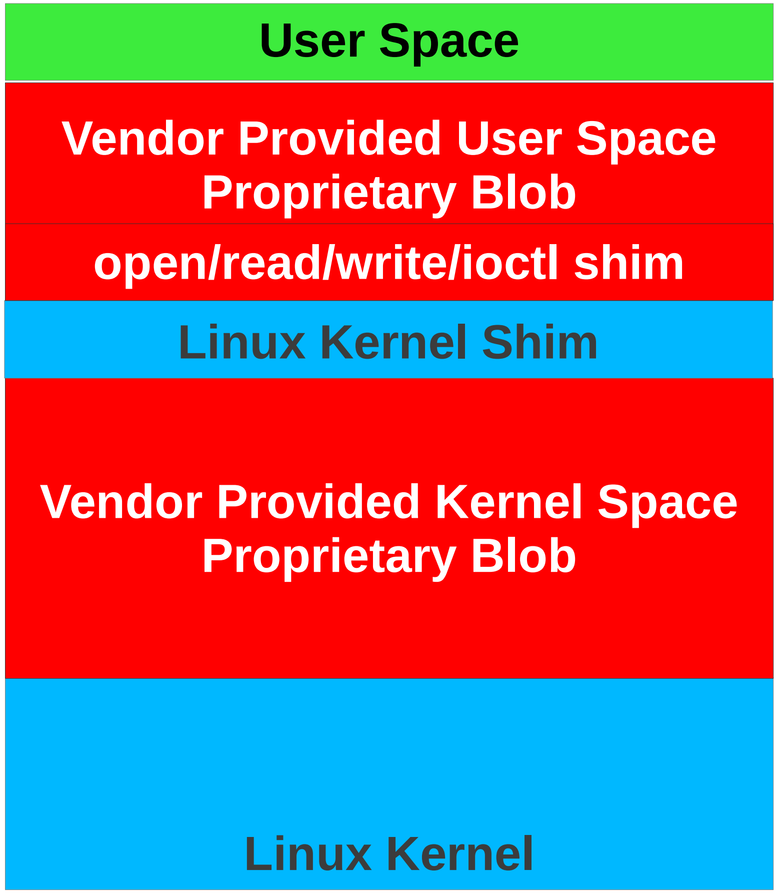
Vendor Provided User Space Proprietary Blob

Well Defined User Space API

Well Defined Kernel Space API

Vendor Provided Kernel Space Proprietary Blob

Typical Proprietary Stacks on Linux



User Space

Vendor Provided User Space Proprietary Blob

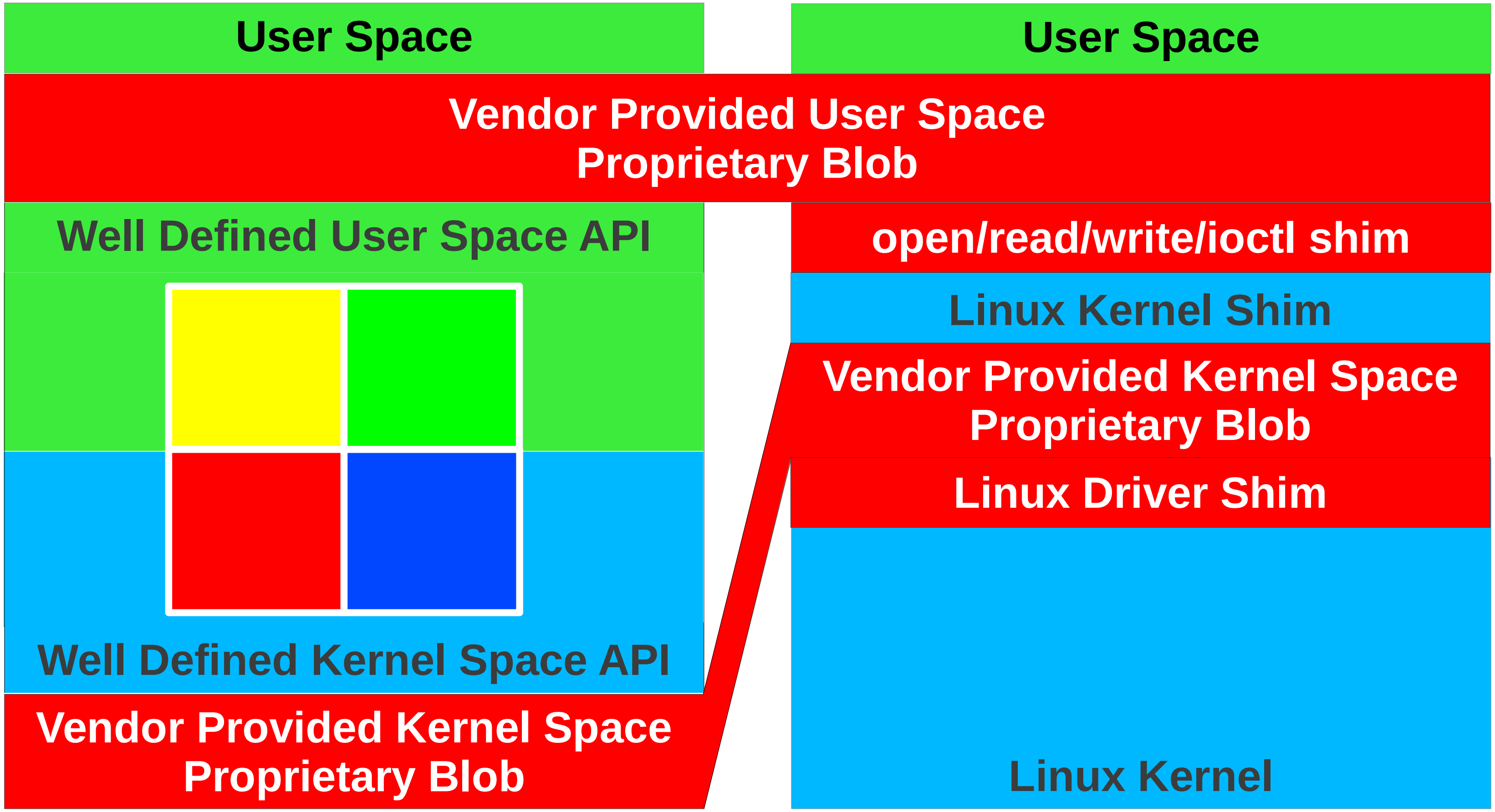
open/read/write/ioctl shim

Linux Kernel Shim

Vendor Provided Kernel Space Proprietary Blob

Linux Kernel

SoC vendors want to write code once and protect IP


User Space
User Space
Vendor Provided User Space Proprietary Blob
Well Defined User Space API
open/read/write/ioctl shim
Linux Kernel Shim
Vendor Provided Kernel Space Proprietary Blob
Linux Driver Shim
Well Defined Kernel Space API
Vendor Provided Kernel Space Proprietary Blob
Linux Kernel

Things we can do

- Examine what crosses the OS barrier and see what we can do to decouple things, such as:
  - Rethink memory and power policy and mechanism at the OS boundary
  - Create official binary box interfaces
  - Split devices from the kernel completely
  - Enable hacking and proprietary solutions

Things we can do

- OEMs and ODMs can work with SoC manufactures to ensure the source flows
- Reverse engineer

Thanks!

Feel free to send questions, comments and rants to zach.pfeffer@linaro.org

or

pfefferz at #linaro-android on irc.freenode.net

or

Zach Pfeffer (not the soccer player) on G+

Binary Blobs Attack!!!

Zach Pfeffer (pfefferz)
Linaro Android Platform Team

Embedded Linux Conference 2012

I'm working on a movie...

Here's the script...

Working Title: Binary Blobs Attack!!!

8 AM - San Jose
Philip
...stretches, pees, starts checking mail

9 AM
Philip
"okay, I need to write a message through this blob to the core"

11 AM
Philip
"Crap! What's wrong, why did you crash?"

1 PM San Jose
2:30 AM Bangalore
Philip
"Yeah, I'm not sure, I send the message and BOOM! The core just stops."
FAE of FaST
"I think that teams in India, I'll send a mail"
Philip
"Okay..."

10 AM Bangalore, the next day

8:30 PM San Jose

Vishal

"I got your mail..."

FAE of FaST

NDA mumble, NDA mumble, NDA mumble...

Vishal

"right, you need to send the version of the interface you want before using it"

FAE of FaST

NDA mumble, NDA mumble, NDA mumble...

Vishal

"no you have the v2.2 of the doc, its on page 34x-rwfv-444-:) of version 2.2.0.1, check Dora for the new version"

11 PM San Jose

FAE of FaST

opens beer

"let me send this before I go to bed..."

8 AM San Jose
9:30 PM Bangalore

Philip

"okay, I sent the version"
runs the unit...
"crap! what's wrong with the graphics stack"
calls Ngraphics' FAE...

A Word Problem

Father's Day is in 13 days. SuperCoolNotGoogle and FaST is working on the FathersDayChecksYourFridgeHasBeer phone. It estimates that it will need to solve 325 crashes. 2 engineers and 1 FAE can solve 1 crash per day. How many Engineers and FAEs does SuperCoolNotGoogle need to ship their product?

325 crashes until launch

13 days until launch

325 crashes / 13 days = 25 crashes / day

2 engineers + 1 FAE = 1 crash / day

Answer

50 engineers + 25 FAEs

My movie wasn't picked up :(

Maybe I'll do a web series...

Here's the web series script...

Working Title: Binary Blobs Attack Again!

8 AM - San Jose

Philip

...stretches, pees, starts checking mail

9 AM

Philip

"okay, I need to write a message through this blob to the core"

11 AM

Philip

"Crap! What's wrong, why did you crash?"

1 PM San Jose

Philip

looks at source, puts a print in at the remote call, checks log
"Hmm... I guess I need to send the version first"
work, work, work
"Bingo!"

Another Word Problem

Mother's Day is in 13 days. SuperCoolNotGoogle is working on the MothersDayRemindsYourKidsToCall phone. It estimates that it will need to solve 325 crashes. SuperCoolNotGoogle has the source to everything so 1 engineer can solve 2 crashes per day. How many Engineers does SuperCoolNotGoogle need to ship their product?

325 crashes until launch

13 days until launch

325 crashes / 13 days = 25 crashes / day

2 engineers = 2 crashes / day

Answer

12.5 engineers

Wow!!!

Having the source allowed SuperCoolNotGoogle and FaST to save 12.5 engineers and 25 FAEs.

Maybe they could get 2 phones out instead of 1 or...

Time spent on binary blob production, integration and debugging has eclipsed any other activity in the product cycle.

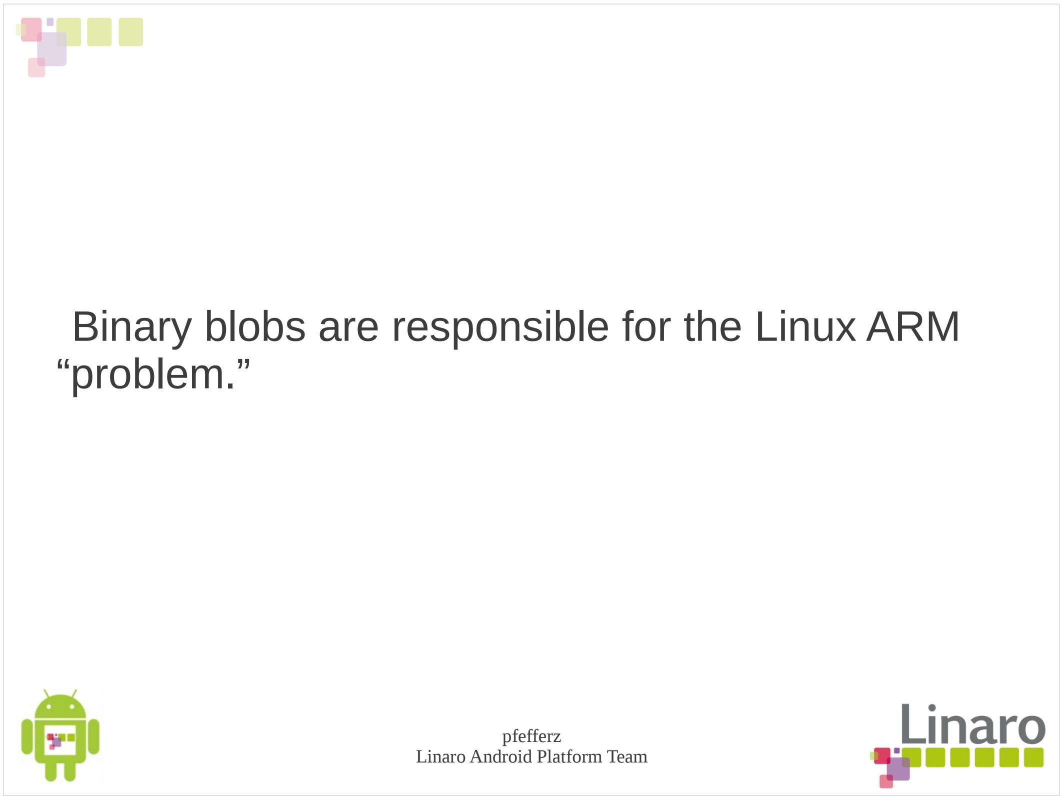
Engineer Rants

- “Binary blob scribbling is impossible to debug”
- “Kernel upgrades mean a recompile cycle with the SoC's 3rd party vendor”
- “Binary blobs are un-ethical”
- “OS independent code is always buggy, slow and will only be used on the OS its not meant for”

- “I need debug symbols!”
- “Documentation, Helloooo”
- “I'm going to kick ARM out of the kernel”
- “I can't track Linus' tip because I'll lose enablement”
- “I need a JTAG”
- "Write once, run everywhere doesn't really work for Java, so how well is that going to work for a driver?"

Problems

- The audio blob is locked to the 3.0 kernel the graphics blob is locked to 3.1
- hardfp and other toolchain improvements are lost
- The blob producers are impossible to track down
- Has brought back “batch” development



Binary blobs are responsible for the Linux ARM “problem.”

pfefferz
Linaro Android Platform Team

Linaro

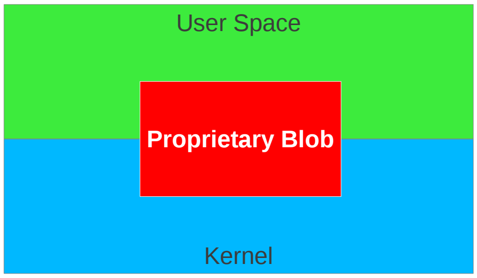
Upstream won't look at shim's, pass throughs or other kernel or userspace blob frontends.

This also leads to an inability to refactor ARM support into common kernel code, further marginalizing ARM

Binary blobs won't be going away.

How can we ensure an open, sustainable software eco-system with them?

Most problems happen when proprietary code crosses the userspace/kernel boundary

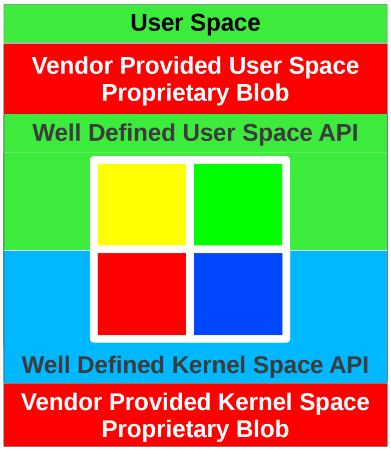


User Space

Proprietary Blob

Kernel

How Windows deals with this (so I've heard)



User Space

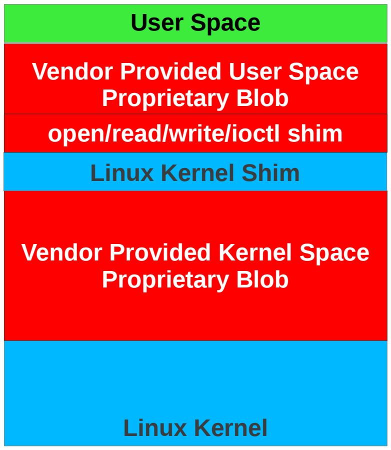
Vendor Provided User Space Proprietary Blob

Well Defined User Space API

Well Defined Kernel Space API

Vendor Provided Kernel Space Proprietary Blob

Typical Proprietary Stacks on Linux



User Space

Vendor Provided User Space Proprietary Blob

open/read/write/ioctl shim

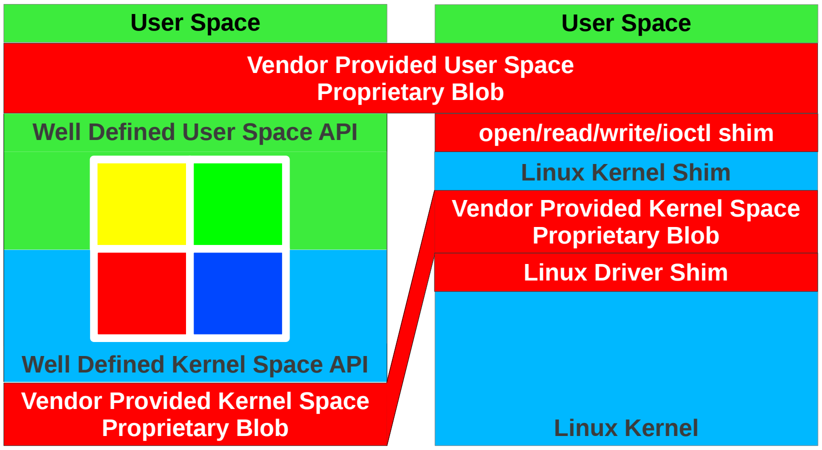
Linux Kernel Shim

Vendor Provided Kernel Space Proprietary Blob

Linux Kernel

pfefferz
Linaro Android Platform Team

SoC vendors want to write code once and protect IP



User Space

User Space

Vendor Provided User Space Proprietary Blob

Well Defined User Space API

open/read/write/ioctl shim

Linux Kernel Shim

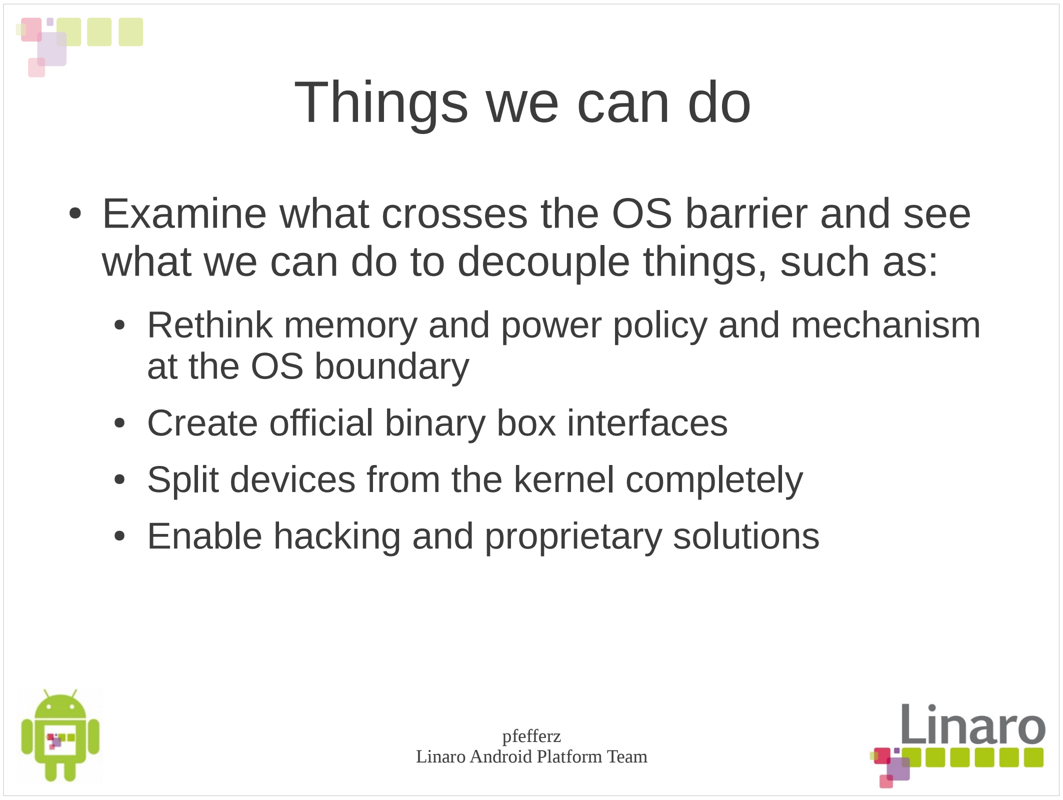
Vendor Provided Kernel Space Proprietary Blob

Linux Driver Shim

Well Defined Kernel Space API

Vendor Provided Kernel Space Proprietary Blob

Linux Kernel



Things we can do

- Examine what crosses the OS barrier and see what we can do to decouple things, such as:
  - Rethink memory and power policy and mechanism at the OS boundary
  - Create official binary box interfaces
  - Split devices from the kernel completely
  - Enable hacking and proprietary solutions

Things we can do

- OEMs and ODMs can work with SoC manufactures to ensure the source flows
- Reverse engineer

Thanks!

Feel free to send questions, comments and rants to zach.pfeffer@linaro.org

or

pfefferz at #linaro-android on irc.freenode.net

or

Zach Pfeffer (not the soccer player) on G+