Introduction to Linux kernel hacking [workshop]

CodeUtsava 1.0 [NIT Raipur]

Vaishali Thakkar

(vaishalithakkar94@gmail..com, @kernel_girl)

Who Am I?

- Linux Kernel developer at Oracle
- Working in kernel security engineering group and memory management
- Interested in many different subsystems of the Linux Kernel
- Associated with the open source internship programs and some community groups

Agenda

- Prerequisites
- Introduction to Open source
- Introduction to version control system and common open source practices
- Process of the Kernel Development
- Practical and demo
- What after sending first patch in the Linux kernel?

Prerequisites

- Session one:
  - Linux-next source code
  - Git
- Session two:
  - Tools:

    apt install vim libncurses5-dev gcc make exuberant-ctags libssl-dev
  - Setting up gitconfig
  - Configuring, Compiling, installing and booting in to the new kernel
  - Process:

    https://github.com/vthakkar1994/Linux-Kernel-Workshop/blob/master/Prerequisites_NITRaipur.md

Session one: part I

Open source: What?

- A Methodology or philosophy that promotoes free distribution and access to a product's design or ideas and implimentation details
  - Open as in free use
  - Open as in access
  - Open as in available for everyone
  - Dozons of License types
  - Modiafiable and usuable for any purpose
  - Often community driven

Open source: Why?

- Give back to community
- Collaorating with the amazing people worldwide
- Education and learning
- Interesting and challenging work
- Fun
- Your portfolio = your CV

Open source: Licences

- Licenses that comply with the Open Source Definition — allows software to be freely used, modified, and shared.
  - MIT:
    Short, simple and permissive. Requires preservation of copyright and license notices. Patent permission not granted.
  - Apache:
    Requires preservation of copyright and license notices. Provides an express grant of patent rights.
  - GPLv3:
    Strong copyleft license. Modified source code must be distributed under the same license.

Open source: How

- Bugs
  - Finding bugs
  - Reporting bugs
  - Fixing bugs

Open source: How

- Bugs
- Testing
  - Using tools
  - Testing of a new feature
  - Performance testing

Open source: How

- Bugs
- Testing
- Documentation
  - Contributing to existing docs
  - Writing blogs
  - Adding new docs

Open source: How

- Bugs
- Testing
- Documentation
- Translation and localization
  - Helping translating docs in local languages
  - Forming local groups and helping to build a local community by knowledge sharing

Open source: How

- Bugs
- Testing
- Documentation
- Translation and localization
- Community involvement
  - Helping others
  - IRC and mailing lists involvement

Open source: How

- Bugs
- Testing
- Documentation
- Translation and localization
- Community involvement

Kernel Basics

Before diving into the next part of a Session I, configure the kernel and start the compilation process.

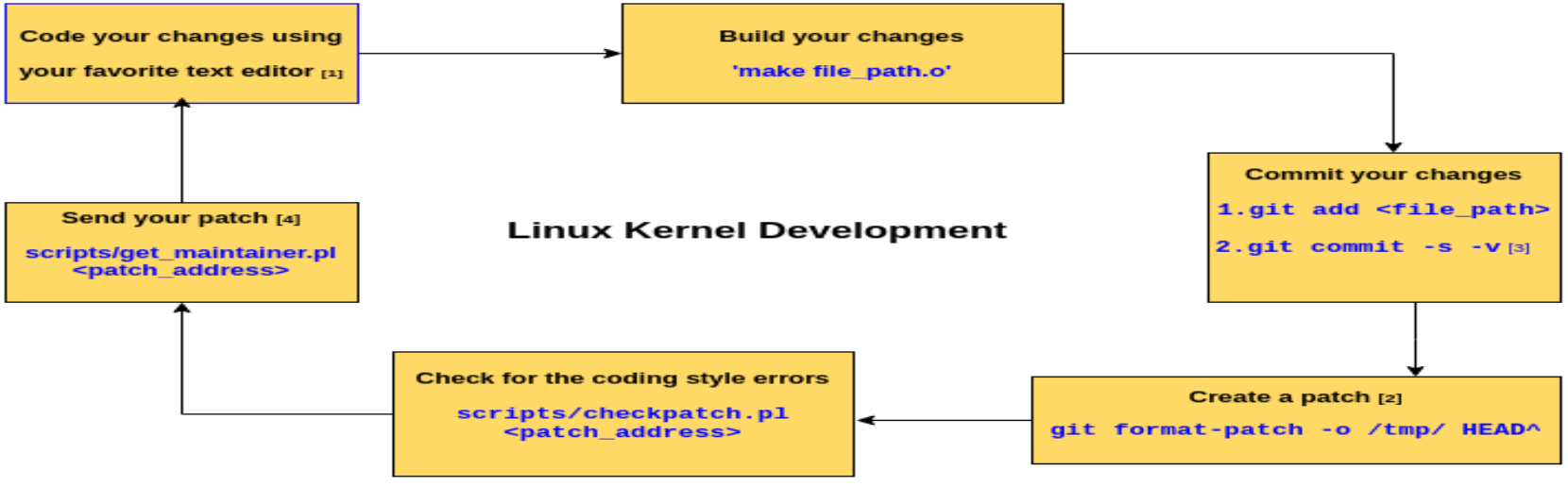
Session one: part II

Process of the kernel development

- Code your changes
  - Find a contribution to make
  - Read mailing list archives
  - Gain experience and ask questions

Process of the kernel development

- Code your changes
- Create and send in your patch



[1] Do the changes in your local branch: https://kernelnewbies.org/FirstKernelPatch#head-4fc0349738a61ed254bcbef7a980321c77495014

[2] You should see the command output with a filename in /tmp/

[3] Philosophy of Linux kernel patches: https://kernelnewbies.org/PatchPhilosophy

[4] Using mutt to send patches: https://kernelnewbies.org/FirstKernelPatch#head-dc6a8aa0be0d0e8ed9dc03726d0b5a1fb0f65e1f
Using git-send-email to send patches: https://burzalodowa.wordpress.com/2013/10/05/how-to-send-patches-with-git-send-email/

Process of the kernel development

- Code your changes
- Create and send in your patch
- Gather feedback
  - Testing results and the patch
  - Mentoring and guidance
  - Discussion of strategies and suggestions

Process of the kernel development

- Code your changes
- Create and send in your patch
- Gather feedback
- Repeat

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
  - Consistent communication style

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
  - Consistent communication style
  - Respond inline - Say NO to top-posting

Example:

```
From: Kludge Crufty <example@email.com>
Subject: Design decisions for next release
On Fri, Sep 12, 2014 at 03:00:56PM -0700, Baaz Quux wrote:
> On Fri, 12 September 2014 at 02:30:17PM -0700, Foo Bar wrote:
>
> I think we should do X.

I think we should do Y.
Kludge
```

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
  - Consistent communication style
  - Respond inline - Say NO to top-posting
  - Make sure to use one of the standard email-client listed Documentation/email-clients.txt

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
  - Consistent communication style
  - Respond inline - Say NO to top-posting
  - Make sure to use one of the standard email-client listed Documentation/email-clients.txt
  - Do not use gmail web interface - line wraps the plain text emails

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
  - Consistent communication style
  - Respond inline - Say NO to top-posting
  - Make sure to use one of the standard email-client listed Documentation/email-clients.txt
  - Do not use gmail web interface - line wraps the plain text emails
  - Do not use Outlook - mangles patches

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
  - Consistent communication style
  - Respond inline - Say NO to top-posting
  - Make sure to use one of the standard email-client listed Documentation/email-clients.txt
  - Do not use gmail web interface - line wraps the plain text emails
  - Do not use Outlook - mangles patches
  - Don't include quotes in your signature

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
- Avoid sending personal emails
  - Likely to be missed

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
- Avoid sending personal emails
  - Likely to be missed
  - Sending email in related mailing list opens door for more perspectives on the same problem

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
- Avoid sending personal emails
- Internet relat chat [IRC]
  - Looks like multi way messaging

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
- Avoid sending personal emails
- Internet relat chat [IRC]
  - Looks like multi way messaging
  - Use a dedicated client [not a web client]

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
- Avoid sending personal emails
- Internet relat chat [IRC]
  - Looks like multi way messaging
  - Use a dedicated client [not a web client]
  - Connect to a network

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
- Avoid sending personal emails
- Internet relat chat [IRC]
  - Looks like multi way messaging
  - Use a dedicated client [not a web client]
  - Connect to a network
  - Once on a network, join a channel

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
- Avoid sending personal emails
- Follow IRC Ettiquetts

Communication style - Linux kernel

- Mailing lists - scripts/get_maintainer.pl
- Responding to emails
- Avoid sending personal emails
- Follow IRC Ettiquetts
- Remember, anything you post on the internet is there FOREVER.

Coding style - Linux kernel

- Documentation/codingstyle

Coding style - Linux kernel

- Documentation/codingstyle
- We love tabs

Coding style - Linux kernel

- Documentation/codingstyle
- We love tabs
- 80 characters line limit

Coding style - Linux kernel

- Documentation/codingstyle
- We love tabs
- 80 characters line limit
- We are picky - checkpatch.pl

Git

- Open source
- Created by Linus Torvalds in 2005 to work on the Linux kernel
- Fastest version control system
- Set up .gitconfig to send patches in the Linux kernel:

Git patch

- Example:

```
clk: sunxi: Use resource_size

Use the function resource_size instaed of explicit computation.
Problem found using Coccinelle.

Signed-off-by: Vaishali Thakkar <vaishali.thakkar@oracle.com>
Acked-by: Chen-Yu Tsai <wens@csie.org>
Signed-off-by: Maxime Ripard <maxime.ripard@free-electrons.com>

diff --git a/drivers/clk/sunxi/clk-sun9i-mmc.c b/drivers/clk/...
index a9b1761..0c2dd02 100644
--- a/drivers/clk/sunxi/clk-sun9i-mmc.c
+++ b/drivers/clk/sunxi/clk-sun9i-mmc.c
@@ -106,7 +106,7 @@ static int sun9i_a80_mmc_config_clk_probe
                     (struct platform_device *pdev)

        ...
-   count = DIV_ROUND_UP((r->end - r->start + 1), SUN9I_MMC_WIDTH);
+   count = DIV_ROUND_UP((resource_size(r)), SUN9I_MMC_WIDTH);
```

Different tags

- Signed-off-by: Indicates that the signer was involved in the development of the patch, or that he/she was in the patch's delivery path.
- Acked-by: Often used by maintainers or core developers of the file. It is a record that the acker has at least reviewed the patch and has indicated acceptance.
- Reported-by: Gives credit to people who find bugs and report them.

Different tags

- Tested-by: Indicates that the patch has been successfully tested (in some environment)
- Reviewed-by: Indicates that the patch has been reviewed and found acceptable according to the Reviewer's Statement
- Suggested-by: Indicates that the patch idea is suggested by the person named and ensures credit to the person for the idea.

Developer's certificate of origin

- By making a contribution to this project, I certify that:
  - The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or
  - The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

Developer's certificate of origin

- By making a contribution to this project, I certify that:
  - The contribution was provided directly to me by some other person who certified above 2 points or and I have not modified it.
  - I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

References: Session I

- **First kernel patch tutorial:**

  https://kernelnewbies.org/FirstKernelPatch

- **Linux Kernel Documentation:**

  https://kernel.org/doc/html/latest/process/submitting-patches.html

- **Process of patch best practices:**

  https://kernelnewbies.org/PatchPhilosophy

Session Two

Exploring the kernel code with git

Sending first patch into the kernel

What after first patch?

- Use bug finding tools [static checkers, dynamic checkers, fuzzers etc]
- Run kmemleak, kasan and other debugging features. Report bugs in the mailing lists.[1]
- Work on devm_functions and their missing uses
- More ideas: http://www.labbott.name/blog/2016/08/15/ideas-for-getting-started-in-the-linux-kernel/
- More advanced project:

  https://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project

scripts/checkpatch.pl

- Written by Andy Whitcroft, Joe Perches
- Checks for basic coding style issues and sometimes for incorrect API usuage
- Preferrable to run it for any new patch submissions
- Things to take care of:
  - Avoid sending 80 characters line warning
  - If you are sending the patch for the simple warnings, send them for the files in staging/next

scripts/checkpatch.pl

- Example output: perl scripts/checkpatch.pl -f <path_to_{directory, file}>

```
CHECK: spaces preferred around that '+' (ctx:VxV)
#1564: FILE: drivers/staging/media/bcm2048/radio-bcm2048.c:1564:
+       BUG_ON((index+4) >= BCM2048_MAX_RDS_RT);
                         ^

drivers/staging/media/bcm2048/radio-bcm2048.c:1539: CHECK: Avoid
crashing the kernel - try using WARN_ON & recovery code rather
than BUG() or BUG_ON()

drivers/staging/media/bcm2048/radio-bcm2048.c:1997: ERROR: Macros
with complex values should be enclosed in parentheses

drivers/staging/media/bcm2048/radio-bcm2048.c:2025: WARNING:
Prefer 'unsigned int' to bare use of 'unsigned'

drivers/staging/media/bcm2048/radio-bcm2048.c:2543: WARNING:
struct v4l2_ioctl_ops should normally be const
```

Sparse

- Written by Linus Torvalds, later maintained by Josh Triplett, Chris Li
- Essentially, sparse is a library that, like a compiler front end, provides convenient access to the abstract syntax tree and typing information of a C program.
- Provides a set of annotations designed to convey semantic information about types.
  - For example, what address space pointers point to or what locks a function acquires or releases.

Sparse

- Installation:
  - From the package manager of your linux distro:

    e.g. sudo apt-get install sparse
  - Manual installation: https://kernelnewbies.org/Sparse
- Running Sparse: make C=2 <path_to_directory>
- Documentation:
  - Wikipedia: https://en.wikipedia.org/wiki/Sparse
  - Kernel Documentation: Documentation/sparse.txt

Sparse

- Example output:

```
drivers/staging/wlan-ng/p80211conv.c:132:25: warning: cast to
restricted __be16

drivers/staging/wlan-ng/p80211conv.c:154:38: warning: incorrect
type in assignment (different base types)
drivers/staging/wlan-ng/p80211conv.c:154:38: expected unsigned
short [unsigned] [usertype] type
drivers/staging/wlan-ng/p80211conv.c:154:38: got restricted
__be16 [usertype] <noident>

drivers/staging/wlan-ng/prism2fw.c:251:15: warning: memset with
byte count of 120000

drivers/staging/lustre/lnet/selftest/rpc.c:764:9: warning: context
imbalance in 'srpc_shutdown_service' - different lock contexts for
basic block
```

Smatch

- Written by Dan Carpenter
- Uses sparse as a C parser.
- Useful for finding many security[use-after-free, buffer overflow, off-by-one, double locks/unlocks, missing locks etc] related and other bugs.

Smatch

- Installation:
  - git clone git://repo.or.cz/smatch.git
  - cd smatch
  - make
- Running Smatch: <path_to_smatch>/smatch_scripts/kchecker --spammy ./
- Documentation:

  https://blogs.oracle.com/linuxkernel/entry/smatch_static_analysis_tool_overview

Smatch

- Example output:

```
drivers/staging/xgifb/vb_setmode.c:3581 XGI_SetGroup2() warn: mask
and shift to zero

drivers/staging/xgifb/vb_setmode.c:5334 XGI_EnableBridge() warn:
we tested 'pVBInfo->VBInfo & 256' before and it was 'true'

drivers/staging/vt6656/rf.c:876 vnt_rf_table_download() error:
memcpy() 'addr1' too small (3 vs 48)

drivers/staging/rts5208/ms.c:2736 ms_build_l2p_tbl() error:
buffer overflow 'ms_start_idx' 17 <= s32max

drivers/staging/rts5208/ms.c:2594 ms_build_l2p_tbl() error: we
previously assumed 'ms_card->segment' could be null(see line 2586)

drivers/staging/rts5208/sd.c:4115 ext_sd_send_cmd_get_rsp() warn:
masked condition '(*ptr + 3 & 30) != 3' is always true.
```

Coccinelle

- Written by Julia Lawall
- Program matching and transformation tool. It can warn you about bugs [report mode] or suggest a fix for the bugs [patch mode].
- Spatch: Coccinelle binary in /usr/bin or /usr/local/bin that invokes the Coccinelle program.
- Semantic Patch Language(SmPL): Not another scripting language, aware of the structure of the C language

Coccinelle

- Coccicheck:
  - One of the targets of the Linux kernel
  - Provides a series of semantic patches written in SmPL and make use of the Coccinelle engine to interpret and complete these tests.
  - Each script has confidence - High, Moderate, Low
  - Can be run with four modes:
    - Patch - lets you fix the issues found
    - report - lets you generate a report
    - context - highlights lines of interest[indicated by -] and their context in a diff-like style.
    - org - generates a report in the Org mode format of Emacs

Coccinelle

- Installation:
  - From the package manager of your linux distro:
    e.g. sudo apt-get install coccinelle
  - Manual installation: https://github.com/coccinelle/coccinelle
- Running coccicheck:
  - All scripts under scripts/coccinelle: make coccicheck MODE=patch
  - On specific directory:make coccicheck MODE=report M=drivers/net/
  - Running specific tests:make coccicheck
    COCCI=scripts/coccinelle/locks/double_lock.cocci MODE=report
- Documentation: Documentation/coccinelle

Coccinelle

- Example output:

```
./security/integrity/ima/ima_template.c:192:29-35: ERROR:
application of sizeof to pointer

./drivers/power/supply/ab8500_charger.c:3676:8-28: ERROR:
Threaded IRQ with no primary handler requested without
IRQF_ONESHOT

./sound/soc/samsung/i2s.c:1269:2-4: ERROR: test of a variable
/field address

./drivers/block/loop.c:736:8-15: ERROR: PTR_ERR applied after
initialization to constant on line 728

./fs/btrfs/send.c:6335:22-39: ERROR: sctx is NULL but
dereferenced.

./drivers/misc/lkdtm_heap.c:38:1-5: ERROR: reference preceded
by free on line 37
```

Conclusion

- **First kernel patch tutorial:**
  https://kernelnewbies.org/FirstKernelPatch
- **LWN Articles:** https://lwn.net/Kernel/
- **Linux Kernel Documentation:**
  Documentation directory in the Linux kernel source code

Welcome to Linux kernel community!

Thank You