Reboot adieu!
Online Linux kernel patching

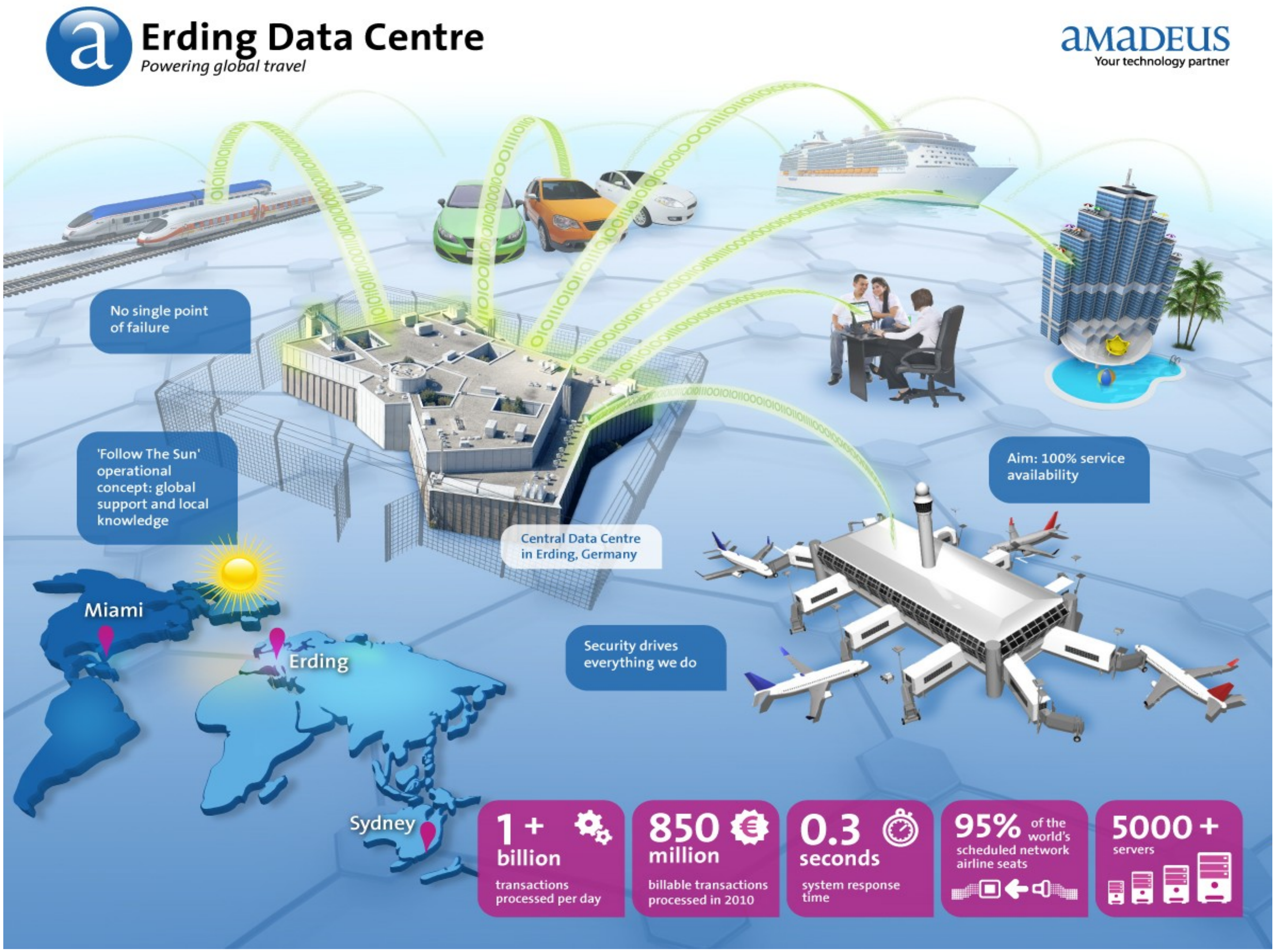
Udo Seidel

Agenda

- Who & Why?
- How?
- Players & Show!
- And?

Me :-)

- Teacher of mathematics and physics
- PhD in experimental physics
- Started with Linux in 1996
- Linux/UNIX trainer
- Solution engineer in HPC and CAx environment
- Head of the Linux Strategy and Server Automation @Amadeus


Erding Data Centre
Powering global travel
amadeus
Your technology partner
No single point of failure
'Follow The Sun' operational concept: global support and local knowledge
Central Data Centre in Erding, Germany
Security drives everything we do
Aim: 100% service availability
Miami
Erding
Sydney
1+ billion transactions processed per day
850 million billable transactions processed in 2010
0.3 seconds system response time
95% of the world's scheduled network airline seats
5000+ servers

BMC Exchange Berlin 2013

Why?

Why kernel updates?

- Business critical applications on Linux
  - Bug fixing
  - New functions
  - improvements
  - External requirements
- Importance of security, e.g. PCI-DSS

What is 'wrong' with reboots?

- Missing HA
- Procedures, Operations, ...
- External requirements

Question ....

Do we really need a reboot?

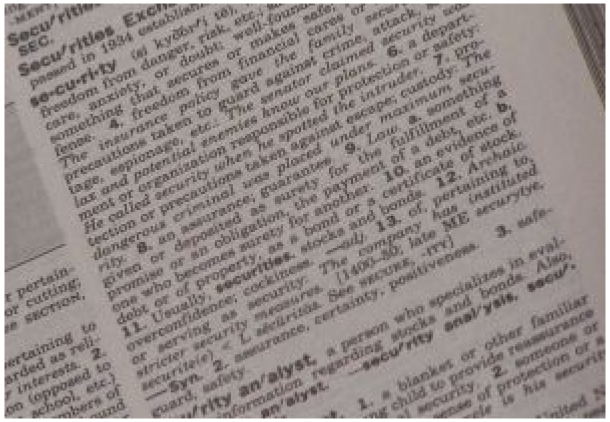
Looking back and around

- Not new
    - mainframes
    - hot updates for Unix
    - Early days of Linux
- Picked up by the 'dark side'
    - Rootkits

How?

Source code comparison

- One approach for generation of hot updates
- Looks simple ... but
  - High programming language skills needed
  - Analysis complex
  - Code replacement unclear



Object code comparison

- Advantages
  - Reduced need for developing skills
  - Implicit patch analysis
  - Can be automated
  - Used already in that context
- Challenges
  - Object code generation
  - Code replacement

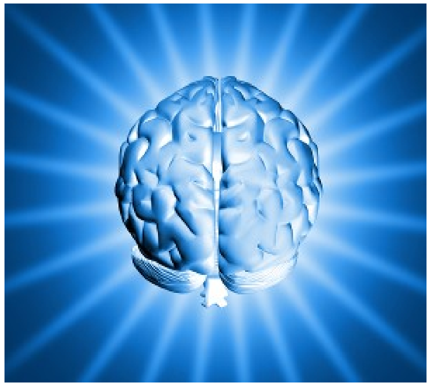
Open questions

- Creation in general
- Detect and cover dependencies
- Activation
- Deactivation(?)
- Management

Players!

Ksplice arises ...

- 2008/2009
  - 4 students @ MIT
    - Thesis from Jeff Arnolds
  - Ksplice Inc. founded
    - GPLv2
    - Supported: Debian, Ubuntu, Fedora, CentOS, RHEL
- July 2011
  - Acquired by Oracle



Ksplice – high level

- Patching original source code
- Generation of new object code
- Comparison of 'old' and new object code
- Load of the delta code
- Address redirection to activate new object code

Ksplice – more details

- See OSTD 2012 ;-)

Ralf Spenneberg Linux Schulungen direkt vom Autor

OpenSource Training

START SCHULUNG BERATUNG UNTERNEHMEN

OpenSource Training > Start > OSTD > 2012

OpenSource Trends Days 2012

The newcomers

- Opensource
  - SUSE
  - Red Hat
- Partially Opensource
  - CloudLinux

Red Hat's kpatch

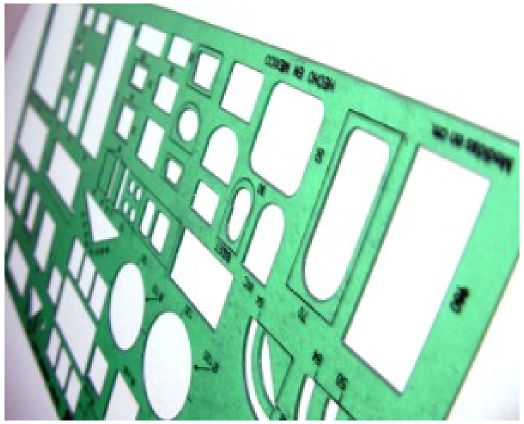
kpatch – History

- Not picked up for a long time
- Big surprise in FEB2014
  - dynup-kpatch project on Github
  - Quite advanced
    - Tools
    - Scripts
    - Documentation

kpatch – How

- Object code comparison
- New function per kernel module
- Jump address manipulation

BUT



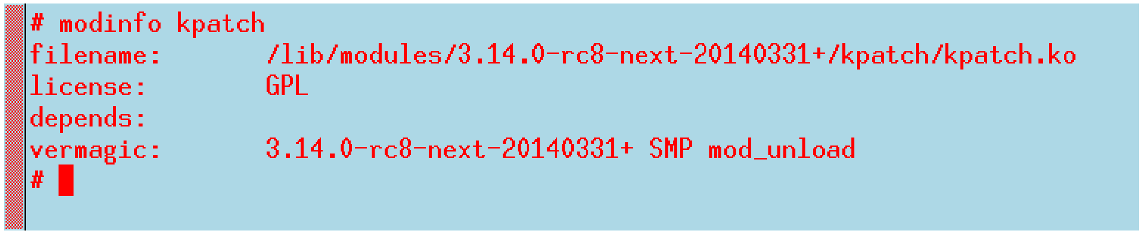
- Out of scope
  - Constantly used system calls
  - virtual Dynamic Share Objects (vDSO)
  - Change (of handling) of data allocation

kpatch – Object Code Comparison

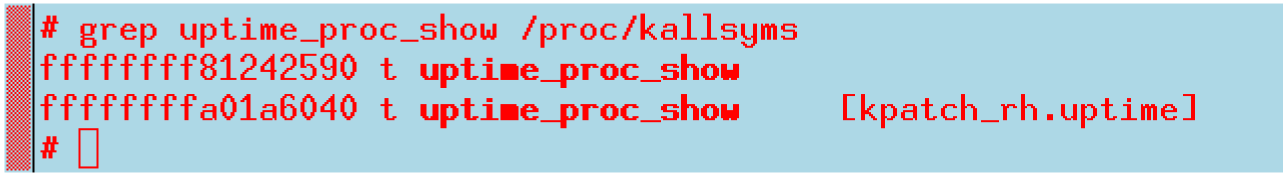
- 2 kernel compilations
  - With & without patch
  - Speedup by CCACHE
  - Compiler flags
    - `-ffunction-sections`
    - `-fdata-sections`
- O/S tools
  - `objcopy`
  - `readelf`

kpatch – New Function to Kernel

- No kernel code change needed
- Two modules
  - Base/Core
  - Patch
- Function name unchanged



```
# modinfo kpatch
filename:       /lib/modules/3.14.0-rc8-next-20140331+/kpatch/kpatch.ko
license:        GPL
depends:
vermagic:       3.14.0-rc8-next-20140331+ SMP mod_unload
#
```



```
# grep uptime_proc_show /proc/kallsyms
ffffffff81242590 t uptime_proc_show
ffffffffa01a6040 t uptime_proc_show     [kpatch_rh.uptime]
#
```

kpatch – Jump Address Challenge

- ftrace
  - *F*unction *TRACE*r
  - `mcount()`
  - kpatch-handler
- stop_machine()
  - Under discussion
  - See Masami Hiramatsu's fork

kpatch – Toolbox

- 2 sets
  - Builder
  - Loader
- http://github.com/dynup/kpatch/

SUSE's kGraft

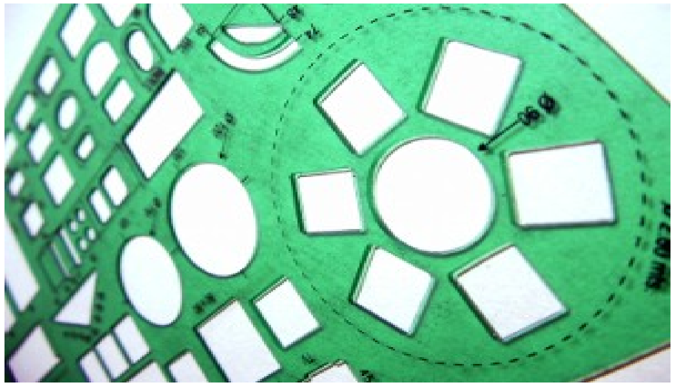
kGraft - History

- 'talks' since spring/summer 2012
  - Announcement at SUSECon 2012
  - Very quiet at SUSECon 2013
- Surprising news in FEB2014
  - Source code not immediately available
  - Presentation at Linux Collaboration Summit
  - Git repository public since MAR2014

kGraft – How

- Object code comparison
- New function per kernel module
- Jump address manipulation

  BUT
- Out of scope
  - Nothing??
  - See later

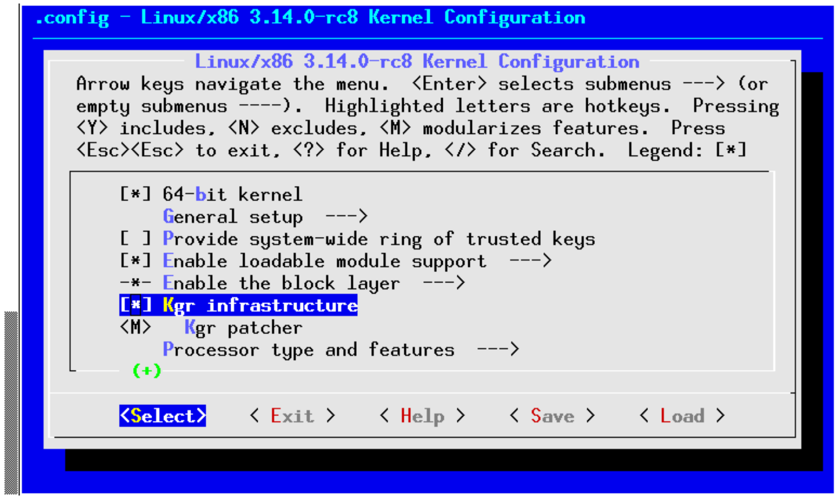


kGraft – Object Code Comparison

- 2 kernel compilations
  - With & without patch
  - Compiler flags
    - `-ffunction-sections`
    - `-fdata-sections`
- O/S tools
  - `objcopy`
  - `nm`
  - `readelf`

kGraft – New Function to Kernel

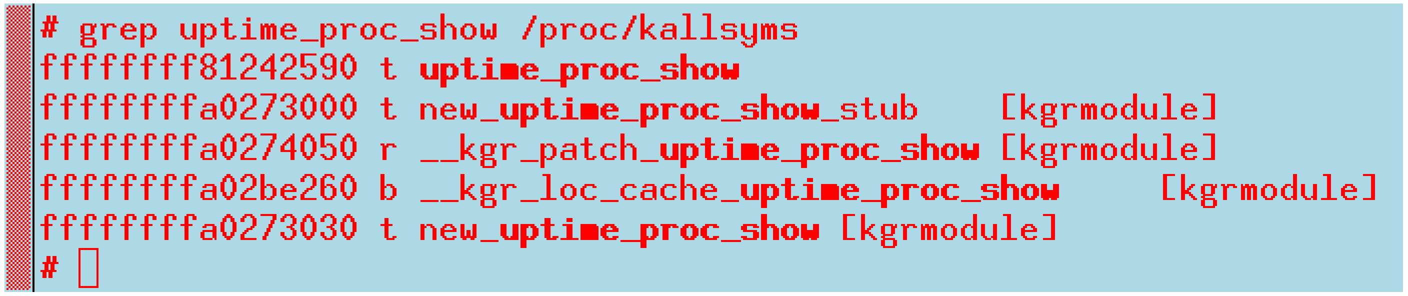
- Requires intra-Kernel infrastructure
  - kernel code change
  - Plan ahead
  - One module
- Function name changed



```
.config - Linux/x86 3.14.0-rc8 Kernel Configuration
              Linux/x86 3.14.0-rc8 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or
  empty submenus ----).  Highlighted letters are hotkeys.  Pressing
  <Y> includes, <N> excludes, <M> modularizes features.  Press
  <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]

      [*] 64-bit kernel
          General setup  --->
      [ ] Provide system-wide ring of trusted keys
      [*] Enable loadable module support  --->
      -*- Enable the block layer  --->
      [*] Kgr infrastructure
      <M>   Kgr patcher
          Processor type and features  --->
       (+)

      <Select>    < Exit >    < Help >    < Save >    < Load >
```



```
# grep uptime_proc_show /proc/kallsyms
ffffffff81242590 t uptime_proc_show
ffffffffa0273000 t new_uptime_proc_show_stub     [kgrmodule]
ffffffffa0274050 r __kgr_patch_uptime_proc_show [kgrmodule]
ffffffffa02be260 b __kgr_loc_cache_uptime_proc_show     [kgrmodule]
ffffffffa0273030 t new_uptime_proc_show [kgrmodule]
# 
```

kGraft – Jump Address Challenge

- Ftrace
  - Similar to kpatch
  - INT3 instruction
- ~~stop_machine()~~
- Reality check function + kernel thread flag
- `schedule_on_each_cpu()`
- `kill/pkill`

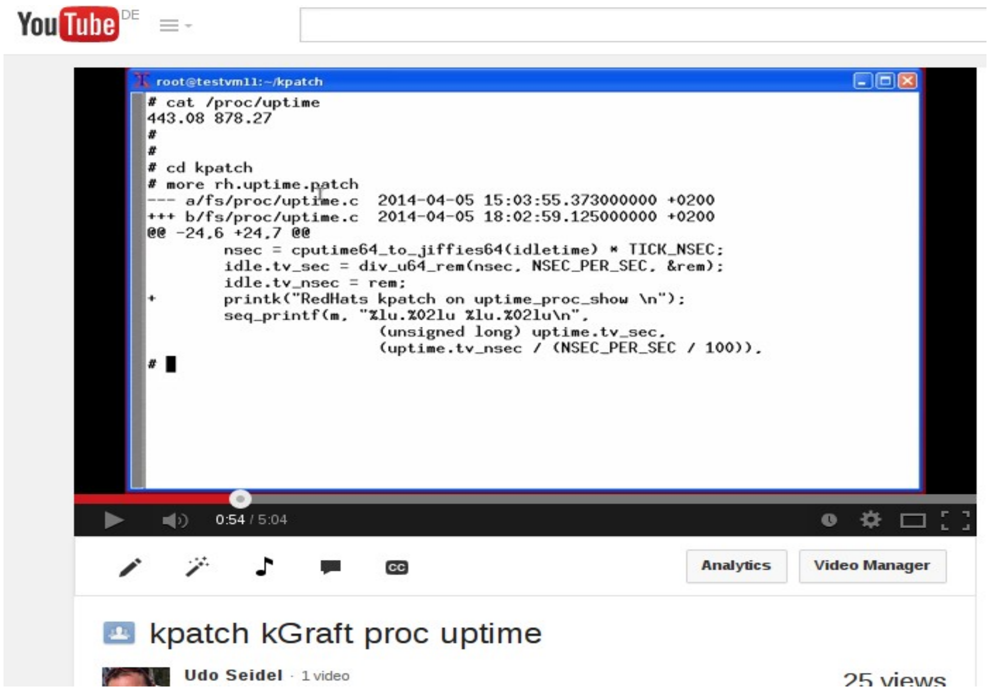
kGraft – Toolbox

- Not really existing
- Part of the kernel
  - Sample code
  - Helper scripts (earlier versions only)
- http://git.kernel.org/cgit/linux/kernel/git/\
  jirislaby/kgraft.git/
- http://github.com/useidel/kgraft-tools

Show!

Example

- uptime_proc_show
- Base: Linux kernel with enabled kGraft
- http://youtu.be/_IQ44jqJdIQ   :-)



```
# cat /proc/uptime
443.08 878.27
#
#
# cd kpatch
# more rh.uptime.patch
--- a/fs/proc/uptime.c  2014-04-05 15:03:55.373000000 +0200
+++ b/fs/proc/uptime.c  2014-04-05 18:02:59.125000000 +0200
@@ -24,6 +24,7 @@
        nsec = cputime64_to_jiffies64(idletime) * TICK_NSEC;
        idle.tv_sec = div_u64_rem(nsec, NSEC_PER_SEC, &rem);
        idle.tv_nsec = rem;
+       printk("RedHats kpatch on uptime_proc_show \n");
        seq_printf(m, "%lu.%02lu %lu.%02lu\n",
                        (unsigned long) uptime.tv_sec,
                        (uptime.tv_nsec / (NSEC_PER_SEC / 100)),
# 
```

kpatch kGraft proc uptime

Udo Seidel · 1 video

25 views

And?

Ongoing

- Patch stacking
  - New/different functions
  - Already patched functions
- Clean patch removal
- Combination with Tracers
  - ftrace
  - Systemtap
  - LTTng

Open items

- Technical
  - 'Highlander' mode
    - Kernel Summit AUG2014
    - Again @ LinuxCon Europe OCT2014
  - Supported architectures
- Enterprise readiness
  - Support
  - Framework
  - RHEL7 & SLES12

Summary

- Both: advantages and disadvantages
- kpatch: more flexible and better tooling
- kGraft: potentially more powerful (?)
- Continued development
- Vanilla Kernel approach (still) unclear
- Keep on watching

References

- http://www.ksplice.com
- http://rhelblog.redhat.com/2014/02/26/kpatch/
- http://www.suse.com/communities/conversations/\
  kgraft-live-kernel-patching/

Thank you!

Reboot adieu!

Online Linux kernel patching

Udo Seidel