Live Patching
A Down in the Trenches View

Sarah Newman
sarah.newman@computer.org
Disclaimers

• Speaker is not “the expert”
Disclaimers

- Speaker is not “the expert”
- Incorrect live patches can cause crashes/ data loss
Agenda

- Terminology
- Why live patching
- Types of live patching
- Building live patches
Terminology

- Hypervisor
- Xen
- CentOS / Ubuntu
- binutils – readelf, objdump
- Program section - .data, .text
- Microcode
Why Live Patching?
Problem Statement

- Apply a software update to kernel or hypervisor (or core userspace program)
What is Live Patching?

- Patching running program – no restarts or reboots required
Alternatives to Live Patching

- Reboot
- Kexec
- Redundant services
- Live migration (virtual machines only)
Why Live Patching

- Not all operating systems are live migration capable
- Live migration is not as well tested
- Live patching is very fast to apply compared to other options
Types of Live Patching
Edit-in-Place

- Overwrite existing function with a new one
- New function must be no larger than existing function
Splicing

- Callers edited to point to new function
Trampoline

- Redirects existing function to new one
- New function can be any size
Building live patches
Implementations

- **Xen**
  - livepatch-build-tools

- **Linux**
  - ksplice - Oracle
  - kpatch - RedHat
  - kgraft - SUSE
  - Canonical live patching - Ubuntu
  - KernelCare - TuxCare
Build Environment

- Should use same compiler, same options as original build
- Easiest to self-compile and preserve original build environment
What can be made into a live patch?

- Easy – logic in function(s), adding variables
- Harder – changing existing data definition
- Hardest – changing size of data structure, initialization code, changing (parts of) livepatch code itself
- Linux and newer versions of Xen can load new microcode at runtime
Will it Patch?

diff --git a/xen/common/schedule.c b/xen/common/schedule.c
index 53daf20..16928728 100644
--- a/xen/common/schedule.c
+++ b/xen/common/schedule.c
@@ -2706,7 +2706,7 @@ void __init scheduler_init(void)
 {
     struct domain *idle_domain;
     int i;
-    printk(KERN_DEBUG "Entering scheduler\n");
+    printk(KERN_DEBUG "Entering scheduler\n");
     scheduler_enable();

     for ( i = 0; i < NUM_SCHEDULERS; i++)
Answer: No

diff --git a/xen/common/schedule.c b/xen/common/schedule.c
index 53daf20f..16928728 100644
--- a/xen/common/schedule.c
+++ b/xen/common/schedule.c
@@ -2706,7 +2706,7 @@ void __init scheduler_init(void)
     }
     struct domain *idle_domain;
     int i;
-
+    printk(KERN_DEBUG "Entering scheduler\n");
     scheduler_enable();
     for ( i = 0; i < NUM_SCHEDULERS; i++)

schedule.o: WARNING: Explicitly ignoring .init section: .init.text
schedule.o: WARNING: Explicitly ignoring .init section: .init.setup
schedule.o: WARNING: Explicitly ignoring .init section: .init.rodata
schedule.o: WARNING: Explicitly ignoring .init section: .init.data
diff --git a/xen/common/schedule.c b/xen/common/schedule.c
index 8ccdb2c4..53daf20f 100644
--- a/xen/common/schedule.c
+++ b/xen/common/schedule.c
@@ -1794,13 +1794,13 @@ long do_set_timer_op(s_time_t timeout)

    return 0;
  }

+long patched = -1;

 /* sched_id - fetch ID of current scheduler */
 int sched_id(void)
 { +
    if (patched < 0) { printk(KERN_DEBUG "patched %ld\n", patched); } patched++;
    return ops.sched_id;
 }

 /* Adjust scheduling parameter for a given domain. */
 long sched_adjust(struct domain *d, struct xen_domctl_scheduler_op *op)
 {
Answer: Yes

diff --git a/xen/common/schedule.c b/xen/common/schedule.c
index 8ccdb2c4..53daf20f 100644
--- a/xen/common/schedule.c
+++ b/xen/common/schedule.c
@@ -1794,13 +1794,13 @@ long do_set_timer_op(s_time_t timeout)

        return 0;
    }

+-long patched = -1;
+/* sched_id - fetch ID of current scheduler */
int sched_id(void)
{
    if (patched < 0) { printk(KERN_DEBUG "patched %ld\n", patched); } patched++;
        return ops.sched_id;
}

/* Adjust scheduling parameter for a given domain. */
long sched_adjust(struct domain *d, struct xen_domctl_scheduler_op *op)
{

(XEN) livepatch: patched: Applying 1 functions
(XEN) livepatch: patched finished APPLY with rc=0
(XEN) patched -1
Will it Patch?
Answer: No

diff --git a/xen/common/sched_credit2.c b/xen/common/sched_credit2.c
index 0aef547a..0a3577c6 100644
--- a/xen/common/sched_credit2.c
+++ b/xen/common/sched_credit2.c
@@ -4103,7 +4103,7 @@ csched2_deinit(struct scheduler *ops)
   xfree(prv);
 }

-static const struct scheduler sched_credit2_def = {
+static struct scheduler sched_credit2_def = {
   .name = "SMP Credit Scheduler rev2",
   .opt_name = "credit2",
   .sched id = XEN SCHEDULER CREDIT2,

ERROR: sched_credit2.o: symbol changed sections: sched_credit2_def,
sched_credit2_def, .data.rel.local.sched_credit2_def, .data.rel.ro.local.sched_credit2_def
Will it Patch?

diff --git a/xen/include/xen/sched-if.h b/xen/include/xen/sched-if.h
index b366f17..e231c822 100644
--- a/xen/include/xen/sched-if.h
+++ b/xen/include/xen/sched-if.h
@@ -333,6 +333,7 @@ struct scheduler {
     struct xen_sysctl_scheduler_op *
   }

   void (+dump_settings) (const struct scheduler *
+  int used;
   void (*dump_cpu_state) (const struct scheduler *, int);
Answer: No

diff --git a/xen/include/xen/sched-if.h b/xen/include/xen/sched-if.h
index b366f177..e231c822 100644
--- a/xen/include/xen/sched-if.h
+++ b/xen/include/xen/sched-if.h
@@ -333,6 +333,7 @@ struct scheduler {
     struct xen_sysctl_scheduler_op *);
     void (*dump_settings) (const struct scheduler *);
     void (*dump_cpu_state) (const struct scheduler *, int);
+    int used;

ERROR: sched_rt.o: object size mismatch: sched_rtds_def
ERROR: sched_credit.o: object size mismatch: sched_credit_def
ERROR: sched_arinc653.o: object size mismatch: sched_arinc653_def
ERROR: schedule.o: object size mismatch: ops
ERROR: sched_null.o: object size mismatch: sched_null_def
ERROR: sched_credit2.o: object size mismatch: sched_credit2_def
Workaround: Hooks

- Apply hook: make changes right before / during / after patch is applied
- Revert hook: make changes right before / during / after patch is reverted
- Used for:
  - Sanity checks
  - Modifying data
Workaround: Shadow Variables

- In Linux, implemented by Shadow Variable API
- In Xen, have to hand-roll
Alternative Instructions

• Different versions of CPUs have different capabilities, different instruction sets
• A single binary must be “lowest common denominator” for instruction set
• Alternative instructions is a framework for patching in more advanced functionality at load time
• Must be run as part of live patching
Patching Xen

https://commons.wikimedia.org/wiki/File:Hyperviseur.png
Consistency Model

- Patches are applied at once to xen/all virtual machines with interrupts disabled
- Probably shouldn't patch anything in the call stack while livepatch code is running (old code will continue to execute)
Process of Applying Patch

- **Upload**
  - Memory allocated
  - Symbols resolved
  - Alternative instructions applied

- **Apply**
  - Work to apply livepatch scheduled
  - CPUs “rendezvous” at specific point in code
  - First CPU there applies patch
    - Run hooks
    - Save old instructions
    - Overwrite instructions with jump
The logic for acquiring a type reference has a race condition, whereby a safety TLB flush is issued too early and creates a window where the guest can re-establish the read/write mapping before writeability is prohibited.

Total of one function patched: _get_page_type in mm.c
Building the Patch

```
# livepatch-build -s xen-RELEASE-4.13.4-9-gce49a1d6d8/ -c config \
-p xsa401-4.13-combined.patch \
   --depends $(readelf --wide --notes xen-syms | grep Build | cut -f2 -d:) \
   --xen-depends $(readelf --wide --notes xen-syms | grep Build | cut -f2 -d:) \
   --xen-syms xen-syms -o xsa401
Building LivePatch patch: xsa401-4-13-combined
Perform full initial build with 8 CPU(s)...  
Reading special section data  
Apply patch and build with 8 CPU(s)... 
Unapply patch and build with 8 CPU(s)... 
Extracting new and modified ELF sections... 
...  
Processing xen/arch/x86/mm.o 
Creating patch module...   
xsa401-4-13-combined.livepatch created successfully
```
Problem 0: bitrot

- centOS 6 + xen 4.8 – worked as of 2020
- centOS 7 + xen 4.14 – doesn't work out of the box
- ubuntu 22.04 + xen 4.16 – doesn't work out of the box
Process of Elimination

- CentOS 7 + xen 4.10, 4.12, 4.13 – can generate
- xen 4.13 security supported through 2022-12-18, good enough for now
- Master + public patches builds but patches don't backport to xen 4.14, 4.16 cleanly
Problems Encountered

• Needed dwarf debug info (gcc -g) re-enabled
  – Added:
    • CONFIG_EXPERT=y or CONFIG_DEBUG=y
    • CONFIG_DEBUG_INFO=y

• create-diff-object segfault (ubuntu 22.04, fixed in kpatch)

• Changed function not detected (ubuntu 22.04)
Try to load...

# xen-livepatch upload xsa401 xsa401-4-13-combined.livepatch
Uploading xsa401-4-13-combined.livepatch... failed
Error 95: Operation not supported

(XEN) livepatch: xsa401: Wrong version (2). Expected 1
Loading, Take 2

```
# livepatch-build -s xen-RELEASE-4.13.4-9-gce49a1d6d8/ \  
   -c config -p xsa401-4.13-combined.patch \  
   --depends $(readelf --wide --notes xen-syms | grep Build | cut -f2 -d:) \  
   --xen-syms xen-syms -o xsa401

# xen-livepatch upload xsa401 xsa401-4.13-combined.livepatch
Uploading xsa401-4.13-combined.livepatch... completed

# xen-livepatch list
```

```
<table>
<thead>
<tr>
<th>ID</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>xsa401</td>
<td>CHECKED</td>
</tr>
</tbody>
</table>
```

```
[root@test ~]# xen-livepatch apply xsa401
Applying xsa401... completed
```
More than expected?

(XEN) livepatch.c:835: livepatch: xsa401: overriding symbol arch/x86/mm.c#get_page_type
(XEN) livepatch.c:835: livepatch: xsa401: overriding symbol new_guest_cr3
(XEN) livepatch.c:835: livepatch: xsa401: overriding symbol mmcfg_intercept_write
(XEN) livepatch.c:835: livepatch: xsa401: overriding symbol donate_page
(XEN) livepatch.c:835: livepatch: xsa401: overriding symbol do_mmu_update
(XEN) livepatch.c:835: livepatch: xsa401: overriding symbol do_mmuext_op
(XEN) livepatch.c:835: livepatch: xsa401: overriding symbol mmio_ro_emulated_write
(XEN) livepatch.c:835: livepatch: xsa401: overriding symbol steal_page
(XEN) livepatch.c:1243: livepatch: xsa401: timeout is 30000000ns
(XEN) livepatch.c:1354: livepatch: xsa401: CPU0 - IPIing the other 1 CPUs
(XEN) livepatch: xsa401: Applying 8 functions
(XEN) livepatch: xsa401 finished APPLY with rc=0
$ objdump --syms xsa401-4-13-combined.livepatch | grep " F " | grep -v UND
0000000000000000 l  F .text._get_page_type  00000000000001a81 arch/x86/mm.c#_get_page_type
0000000000000000 g  F .text.new_guest_cr3  00000000000002a5 .hidden new_guest_cr3
0000000000000000 g  F .text.mmcfg_intercept_write  0000000000000c9 .hidden mmcfg_intercept_write
0000000000000000 g  F .text.donate_page  000000000000021a .hidden donate_page
0000000000000000 g  F .text.do_mmu_update  000000000001695 .hidden do_mmu_update
0000000000000000 g  F .text.do_mmuext_op  000000000001738 .hidden do_mmuext_op
0000000000000000 g  F .text.mmio_ro_emulated_write  00000000000066 .hidden mmio_ro_emulated_write
0000000000000000 g  F .text.steal_page  000000000000290 .hidden steal_page
Culprit: log messages?

3296       default: →
3297       → gdprintk(XENLOG_WARNING, "Error while installing new compat.baseptr.%n", PRI_mfn."
3299       mfn_x(mfn));
3300       → return rc; ←
3364       default: →
3365       → gdprintk(XENLOG_WARNING, "Error while installing new compat.baseptr.%n", PRI_mfn."
3367       mfn_x(mfn));
3368       → return rc; ←
New objdump

```bash
# objdump --syms xsa401-4-13-combined-whitespace.livepatch | grep " F " | grep -v UND
0000000000000000 | F .text._get_page_type 00000000000001a81
    arch/x86/mm.c#_get_page_type
```
New patching messages

(XEN) livepatch.c:262: livepatch: xsa401: Resolved old address arch/x86/mm.c#_get_page_type=> ffff82d080291f3e
(XEN) alt table ffff82d08060ca24 -> ffff82d08060cbc8
(XEN) livepatch.c:835: livepatch: xsa401: overriding symbol arch/x86/mm.c#_get_page_type
(XEN) livepatch.c:1243: livepatch: xsa401: timeout is 30000000ns
(XEN) livepatch.c:1354: livepatch: xsa401: CPU1 - IPIing the other 1 CPUs
(XEN) livepatch: xsa401: Applying 1 functions
(XEN) livepatch: xsa401 finished APPLY with rc=0
Other Differences

- On CentOS 6, compiler randomly changed nops in code resulting in erroneously changed functions
- Have not yet observed on CentOS 7
Payload Limit

# Is -lh whitespace/whitespace.livepatch
-rwxr-xr-x 1 root root 2.4M Jul  3 19:37 whitespace/whitespace.livepatch

# xen-livepatch upload whitespace whitespace.livepatch
Uploading whitespace.livepatch... failed
Error 22: Invalid argument

/* Arbitrary limit for payload size and .bss section size. */
#define LIVEPATCH_MAX_SIZE MB(2)

static int verify_payload(const struct xen_sysctl_livepatch_upload *upload, char *n)
{
    if ( get_name(&upload->name, n) )
        return -EINVAL;

    if ( !upload->size )
        return -EINVAL;

    if ( upload->size > LIVEPATCH_MAX_SIZE )
        return -EINVAL;

    if ( !guest_handle_okay(upload->payload, upload->size) )
        return -EFAULT;

    return 0;
}
Consistency Model

- Patches are applied per-task
- Kernel checks stack of sleeping task, with HAVE_RELIABLE_STACKTRACE
- Otherwise, task switched when it returns to user space

Test Patch – modify /proc/devices

--- linux-5.4.0/fs/proc/devices.c  2022-07-19 04:59:11.656164735 +0000
+++ linux-5.4.0/fs/proc/devices.c  2022-07-19 04:59:31.451858780 +0000
@@ -7,7 +7,7 @@
static int devinfo_show(struct seq_file *f, void *v)
{
    int i = *(loff_t *) v;
-
+    if (i == 0) seq_puts(f, "Devices:\n");
    if (i < CHRDEV_MAJOR_MAX) {
        if (i == 0)
            seq_puts(f, "Character devices:\n");
Kpatch - Ubuntu 20.04

/usr/bin/kpatch-build \ 
   --debug \ 
   --vmlinux /usr/lib/debug/boot/vmlinux-5.4.0-121-generic \ 
   --sourcedir linux-5.4.0/ \ 
   --config config-5.4.0-121-generic \ 
   devices.patch

+CPU   %MEM    TIME+  COMMAND
100.0  0.1   834:57.69 create-diff-obj
(gdb) bt
#0  0x000007fae6bc61fa9 in __vfsscanf_internal (s=<optimized out>, format=<
optimized out>), argptr=argptr@entry=0x7fff6ea986e0, mode_flags=mode_flags@entry=2)
at vfscanf-internal.c:630
#1  0x000007fae6bc6123d in __isoc99_fscanf (stream=stream@entry=0x55f2d52a9210, format=format@entry=0x55f2d3f9f2f9 "%%x %%s %%s %%s\n") at isoc99_fscanf.c:30
#2  0x0000055f2d3f9ae3a in symvers_read (path=path@entry=0x7fff6ea995ea "/home/sarah/.kpatch/tmp/build-generic/Module.symvers"
, table=<optimized out>, table=<optimized out>) at lookup.c:293
#3  0x0000055f2d3f9b03e in lookup_open (symtab_path=0x7fff6ea995bf "/home/sarah/.kpatch/vmlinux.symtab",
symvers_path=0x7fff6ea995ea "/home/sarah/.kpatch/tmp/build-generic/Module.symvers",
hint=0x55f2d533d4f0 "version.c", locals=0x55f2d533d510) at lookup.c:332
#4  0x0000055f2d3f93c33 in main (argc=<optimized out>, argv=<optimized out
>)
at create-diff-object.c:3496
Kpatch - Ubuntu 20.04

• Offending line:
  ```c
  while (fscanf(file, "%x %s %s %s\n",
               &crc, name, mod, export) != EOF)
    table->exp_nr++;
  ```
Kpatch - Ubuntu 20.04

```
/usr/local/bin/kpatch-build \
  --debug \
  --vmlinux /usr/lib/debug/boot/vmlinux-5.4.0-121-generic \
  --sourcedir linux-5.4.0/ \
  --config config-5.4.0-121-generic \
  devices.patch
```

```
[ 706.784218] livepatch_version: disagrees about version of symbol module_layout
```

Root cause: apt-get source got latest linux source, not -121-generic
Kpatch - Ubuntu 20.04

$ /usr/local/bin/kpatch-build \  
   --vmlinux /usr/lib/debug/boot/vmlinux-5.4.0-122-generic \  
   --sourcedir linux-5.4.0/ --config config-5.4.0-122-generic \  
   --target vmlinux devices.patch  
Using source directory at /home/sarah/projects/scale19x/ubuntu20/linux-5.4.0  
Testing patch file(s)  
Reading special section data  
Building original source  
Building patched source  
Extracting new and modified ELF sections  
devices.o: changed function: devinfo_show  
Patched objects: vmlinux  
Building patch module: livepatch-devices.ko  
SUCCESS
user@ubuntu20:~$ sudo kpatch load livepatch-devices.ko
loading patch module: livepatch-devices.ko
waiting (up to 15 seconds) for patch transition to complete...
transition complete (1 seconds)
user@ubuntu20:~$ head /proc/devices
Devices:
Character devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  5 ttyprintk
Summary

• Live patching is very fast compared to other software update options
• Live patching is performed via a trampoline
• Live patches may need to be specially coded
• Don't assume livepatch tooling will work with any software or compiler version
Special thanks to...

- Alison Chaiken
- Vadim Arshanskiy
- Luke Crawford

For early feedback on this presentation!
References

- https://github.com/dynup/kpatch#patch-author-guide
- Xen 4.14+ live patching:
  https://lore.kernel.org/xen-devel/20220302142711.38953-1-roger.pau@citrix.com/
  and you will need to backport
  4267a33b19d43c988fd4535093c426aa2aec70a1 and
  6ff9a7e62b8c43fe3e9d360fbd49d5854787bc39