Live Update: The Making of

Cristiano Giuffrida  Anton Kuijsten  Andrew S. Tanenbaum

Vrije Universiteit Amsterdam

MINIXCon 2016

Amsterdam, The Netherlands
February 1st, 2016
If We Functioned Like Computers...
"If you think database patching is onerous, then try patching a SCADA system that’s running a power plant."

Kelly Jackson Higgins on the SCADA patch problem, 2013

Solution 1: “Spare time for downtime”
“In one of the biggest computer errors in banking history, Chemical Bank mistakenly deducted about $15 million from more than 100,000 customers’ accounts.”

*Saul Hansell, New York Times, 1994*

**Solution 2: “Roll your upgrades”**
"Our research shows that 75% of successful attacks occur against previously known vulnerabilities for which a remediation was already available."

Neil MacDonald, Gartner Research, 2012

Solution 3: “Don’t patch, don’t tell”
"All problems in computer science can be solved by another level of indirection—but that will usually create another problem."

Butler Lampson, quoting David Wheeler

Our solution: “Live update”
Live Update in the Real World

**Servers protected with Ksplice Uptrack:**
100,000+ at more than 700 companies

**Updates applied on production systems:**
More than 2 million and counting

**How it works**

Your Linux vendor releases an update.

Ksplice converts the update into a rebootless update.

You install the update seamlessly, without rebooting.

**Source:** http://www.ksplice.com
--- a/drivers/md/dm-crypt.c
+++ b/drivers/md/dm-crypt.c
@@ -690,6 +690,8 @@ bad3:
    crypto_free_tfm(tfm);
    bad1:
+   /* Must zero key material before freeing */
+   memset(cc, 0, sizeof(*cc) + cc->key.size * sizeof(u8));
    kfree(cc);
 return -EINVAL;
}
@@ -706,6 +708,9 @@ static void crypt_dtr(...)
    cc->iv_gen_ops->dtr(cc);
    crypto_free_tfm(cc->tfm);
    dm_put_device(ti, cc->dev);
+   /* Must zero key material before freeing */
+   memset(cc, 0, sizeof(*cc) + cc->key.size * sizeof(u8));
    kfree(cc);
}

Linux kernel security patch for CVE-2006-0095
Existing Live Update Solutions for C

**Safe update state**
- Update-agnostic characterization, e.g., no updates to active code.
- Problem: extensive patch inspection required for update safety.

**State transfer**
- Automatic generation of basic type transformers.
- Problem: significant programming effort for complex updates.

**Live update mechanisms**
- In-place “hot patching” update strategy.
- Problem: unstable live update process.
Support for simple and complex updates of different natures.

Safe and predictable live update process.

Automated state transfer and state checking.

Automatic error recovery (hot rollback).

Stable live update process.
Our Live Update Design

Process-level updates
Our Live Update Design

Compiler-based state instrumentation

Original Component

Data
Code

Before Instrumentation

Statically Instrumented Component

Data
Metadata
Instrumented code
State management lib

After Instrumentation

Compiler-based state instrumentation
Our Live Update Design

Controlled live update transaction
PROTEOS Architecture

User applications

- Proc Mgr
- Mem Mgr
- Sched
- New version
- Storage
- Network
- Old version
- Upd Mgr
- Disk Driver
- NIC Driver
- KBD Driver
- ...

IPC Microkernel Hw interface

Live Update: The Making of...
static int my_init() {
  ... //initialization code
  return 0;
}
int main() {
  event_eh_t my_ehs = {init : my_init};
  sys_startup(&my_ehs);
  while(1) { // event loop
    msg_t m;
    sys_receive(&m);
    process_msg(&m);
  }
  return 0;
}

Event-driven model
static int my_init() {
    ...  // initialization code
    return 0;
}

int main() {
    event_eh_t my_ehs = {init : my_init};
    sys_startup(&my_ehs);
    while(1) {  // event loop
        msg_t m;
        sys_receive(&m);
        process_msg(&m);
    }
    return 0;
}
static int my_init() {
  ... // initialization code
  return 0;
}

int main() {
  event_eh_t my_ehs = {init : my_init};
  sys_startup(&my_ehs);
  while(1) { // event loop
    msg_t m;
    sys_receive(&m);
    process_msg(&m);
  }
  return 0;
}
Live Update Example

% prctl mupdate `net` /bin/net.new \
    -state 'num_pending_writes == 0'

% prctl mupdate `e1000` /bin/e1000.new

% prctl mupdate-start

UM: Live update requested for net, e1000.
UM: Loading /bin/net.new in memory...
UM: Loading /bin/e1000.new in memory...
UM: Applying changes...
UM: Cleaning up old version...
UM: Live update done.

Multi-component live update
Live Update Example

% prctl mupdate net /bin/net.new \n   -state 'num_pending_writes == 0'

% prctl mupdate e1000 /bin/e1000.new

% prctl mupdate-start

UM: Live update requested for net, e1000.
UM: Loading /bin/net.new in memory...
UM: Loading /bin/e1000.new in memory...
UM: Applying changes...
UM: Cleaning up old version...
UM: Live update done.

State filter
Live Update Example

% prctl mupdate net /bin/net.new \ 
-\ state 'num_pending_writes == 0'

% prctl mupdate e1000 /bin/e1000.new

% prctl mupdate-start

UM: Live update requested for net, e1000.
UM: Loading /bin/net.new in memory...
UM: Loading /bin/e1000.new in memory...
UM: Applying changes...
UM: Cleaning up old version...
UM: Live update done.

Changes applied automatically
The Live Update Process

Update Manager

V1
State
Metadata

V2
State
Metadata
The Live Update Process

Update Manager

PREPARE

V1
State
Metadata

V2
State
Metadata

Cristiano Giuffrida
The Live Update Process

Update Manager

V1
State
Metadata
Update point

V2
State
Metadata

PREPARE READY

Live Update: The Making of Cristiano Giuffrida
The Live Update Process

Update Manager

V1
State
Metadata
Update point

V2
State
Metadata

INIT

Live Update: The Making of Cristiano Giuffrida
The Live Update Process

V1
State
Metadata

Update point

IPC-based
Metadata migration

V2
State
Metadata

TRACE

Update Manager
The Live Update Process
The Live Update Process

Update Manager

V1
State
Metadata
Update point

V2
State
Metadata
Update point

Control flow migration

TRACE
The Live Update Process

Live Update: The Making of

V1
State
Metadata
Update point

CLEANUP

Update Manager

V2
State
Metadata
The Live Update Process

Update Manager
struct s { //old version
    int flags;
    char string[3];
    short id;
    union IXFER(my_u) u;
    void *userdata;
    PXFER(int) addr;
} my_s;
short *p = &my_s.id;

struct s { //new version
    int flags;
    int id;
    char string[2];
    union IXFER(my_u) u;
    PXFER(int) addr;
    int newfield;
} my_s;
int *p = &my_s.id;
Experience

- Applied 50 real updates (∼15000 LOC) with only 265 ST LOC.
- Written 14 state annotations and 4 state filters.
- Median patch size is more than 10x higher than Ksplice’s.
- Instrumentation cost isolated in allocator operations (1.06-2.30x).
- Instrumentation yields a modest memory overhead (∼0.35x).
Update Time

![Graph showing the relationship between state size (KBytes) and update time (ms). The graph plots a curve that indicates a linear increase in update time as the state size increases.]

Live Update: The Making of 

Cristiano Giuffrida
Proteos: a new research OS designed with live update in mind.

Supports several classes of updates with minimal manual effort.

Full control over the live update transaction.

Simple and stable live update process.

Automated and extensible state transfer and state checking.

State transfer error detection and recovery using hot rollback.
Live Update: The Making of

Cristiano Giuffrida, Anton Kuijsten, Andy Tanenbaum
{giuffrida,kuijsten,ast}@cs.vu.nl

Vrije Universiteit Amsterdam