

[patch 6/6] statistics infrastructure – exploitation: zfc

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-12/msg04377.html>

- *From:* Martin Peschke <mp3@xxxxxxxxxxx>
 - *Date:* Wed, 14 Dec 2005 17:14:30 +0100
-

[patch 6/6] statistics infrastructure - exploitation: zfc

This patch instruments the zfc driver and makes it feed statistics data into the statistics infrastructure.

Signed-off-by: Martin Peschke <mp3@xxxxxxxxxxx>
Acked-by: Andreas Herrmann <aherrman@xxxxxxxxxxx>

```
Makefile | 4 -
zfc_aux.c | 22 ++++++
zfc_ccw.c | 7 ++
zfc_def.h | 31 ++++++++
zfc_erp.c | 4 +
zfc_ext.h | 6 +
zfc_fsf.c | 60 ++++++++
zfc_qdio.c | 5 +
zfc_scsi.c | 14 ++++
zfc_stat.c | 195 ++++++++
10 files changed, 338 insertions(+), 10 deletions(-)
```

```
diff -Nurp f/drivers/s390/scsi/Makefile g/drivers/s390/scsi/Makefile
--- f/drivers/s390/scsi/Makefile      2005-10-28 02:02:08.000000000 +0200
+++ g/drivers/s390/scsi/Makefile      2005-12-14 16:01:55.000000000 +0100
@@ -3,7 +3,7 @@
#
```

```
zfc-objs := zfc_aux.o zfc_ccw.o zfc_scsi.o zfc_erp.o zfc_qdio.o \
-          zfc_fsf.o zfc_dbf.o zfc_sysfs_adapter.o zfc_sysfs_port.o \
-          zfc_sysfs_unit.o zfc_sysfs_driver.o
+          zfc_fsf.o zfc_dbf.o zfc_stat.o zfc_sysfs_adapter.o \
+          zfc_sysfs_port.o zfc_sysfs_unit.o zfc_sysfs_driver.o
```

[patch 6/6] statistics infrastructure – exploitation: zfcps

```
obj-$(CONFIG_ZFCP) += zfcps.o
diff -Nurp f/drivers/s390/scsi/zfcps_aux.c g/drivers/s390/scsi/zfcps_aux.c
--- f/drivers/s390/scsi/zfcps_aux.c      2005-12-14 12:51:26.000000000 +0100
+++ g/drivers/s390/scsi/zfcps_aux.c      2005-12-14 16:01:55.000000000 +0100
@@ -778,15 +778,20 @@ zfcps_unit_enqueue(struct zfcps_port *port
     unit->sysfs_device.release = zfcps_sysfs_unit_release;
     dev_set_drvdata(&unit->sysfs_device, unit);

+
+     if (zfcps_unit_statistic_register(unit))
+         return NULL;
+
     /* mark unit unusable as long as sysfs registration is not complete */
     atomic_set_mask(ZFCPS_STATUS_COMMON_REMOVE, &unit->status);

+
+     if (device_register(&unit->sysfs_device)) {
+         zfcps_unit_statistic_unregister(unit);
+         kfree(unit);
+         return NULL;
+     }

+
+     if (zfcps_sysfs_unit_create_files(&unit->sysfs_device)) {
+         zfcps_unit_statistic_unregister(unit);
+         device_unregister(&unit->sysfs_device);
+         return NULL;
+     }
@@ -826,6 +831,7 @@ zfcps_unit_dequeue(struct zfcps_unit *unit
     list_del(&unit->list);
     write_unlock_irq(&zfcps_data.config_lock);
     unit->port->units--;
+
+     zfcps_unit_statistic_unregister(unit);
+     zfcps_port_put(unit->port);
+     zfcps_sysfs_unit_remove_files(&unit->sysfs_device);
+     device_unregister(&unit->sysfs_device);
@@ -837,6 +843,16 @@ zfcps_mempool_alloc(gfp_t gfp_mask, void
     return kmalloc((size_t) size, gfp_mask);
 }

+static void *
+zfcps_mempool_alloc_fsf_req_scsi(unsigned int __nocast gfp_mask, void *data)
+{
+     struct zfcps_adapter *adapter = (struct zfcps_adapter *)data;
+     void *ptr = kmalloc(sizeof(struct zfcps_fsf_req_pool_element), gfp_mask);
+     if (!ptr)
+         statistic_inc(adapter->stat_low_mem_scsi, 0);
+     return ptr;
+}
+
+static void
+zfcps_mempool_free(void *element, void *size)
```

[patch 6/6] statistics infrastructure – exploitation: zfc

```

{
@@ -864,8 +880,8 @@ zfc_allocate_low_mem_buffers(struct zfc

    adapter->pool.fsf_req_scsi =
        mempool_create(ZFCP_POOL_FSF_REQ_SCSI_NR,
-           zfc_mempool_alloc, zfc_mempool_free, (void *)
-           sizeof(struct zfc_fsf_req_pool_element));
+           zfc_mempool_alloc_fsf_req_scsi,
+           zfc_mempool_free, (void *)adapter);

    if (NULL == adapter->pool.fsf_req_scsi)
        return -ENOMEM;
diff -Nurp f/drivers/s390/scsi/zfc_ccw.c g/drivers/s390/scsi/zfc_ccw.c
--- f/drivers/s390/scsi/zfc_ccw.c      2005-10-28 02:02:08.000000000 +0200
+++ g/drivers/s390/scsi/zfc_ccw.c      2005-12-14 16:01:55.000000000 +0100
@@ -163,6 +163,10 @@ zfc_ccw_set_online(struct ccw_device *c
    retval = zfc_adapter_debug_register(adapter);
    if (retval)
        goto out;
+   retval = zfc_adapter_statistic_register(adapter);
+   if (retval)
+       goto out_stat_create;
+
    retval = zfc_erp_thread_setup(adapter);
    if (retval) {
        ZFCP_LOG_INFO("error: start of error recovery thread for "
@@ -183,6 +187,8 @@ zfc_ccw_set_online(struct ccw_device *c
    out_scsi_register:
        zfc_erp_thread_kill(adapter);
    out_erp_thread:
+   zfc_adapter_statistic_unregister(adapter);
+ out_stat_create:
        zfc_adapter_debug_unregister(adapter);
    out:
        up(&zfc_data.config_sema);
@@ -209,6 +215,7 @@ zfc_ccw_set_offline(struct ccw_device *
    zfc_erp_wait(adapter);
    zfc_adapter_scsi_unregister(adapter);
    zfc_erp_thread_kill(adapter);
+   zfc_adapter_statistic_unregister(adapter);
+   zfc_adapter_debug_unregister(adapter);
    up(&zfc_data.config_sema);
    return 0;
diff -Nurp f/drivers/s390/scsi/zfc_def.h g/drivers/s390/scsi/zfc_def.h
--- f/drivers/s390/scsi/zfc_def.h      2005-10-28 02:02:08.000000000 +0200
+++ g/drivers/s390/scsi/zfc_def.h      2005-12-14 16:01:55.000000000 +0100
@@ -58,6 +58,8 @@
    #include <asm/qdio.h>
    #include <asm/debug.h>
    #include <asm/ebcdic.h>
+   #include <asm/timex.h>
+   #include <linux/statistic.h>
    #include <linux/mempool.h>
    #include <linux/syscalls.h>
    #include <linux/ioctl.h>
@@ -66,7 +68,7 @@
    /***** GENERAL DEFINES *****/

```

[patch 6/6] statistics infrastructure – exploitation: zfcpl

```
/* zfcpl version number, it consists of major, minor, and patch-level number */
-#define ZFCPL_VERSION          "4.5.0"
+#define ZFCPL_VERSION          "4.6.0"

/**
 * zfcpl_sg_to_address - determine kernel address from struct scatterlist
@@ -978,6 +980,12 @@ struct zfcpl_adapter {
    struct zfcpl_adapter_mempool    pool;          /* Adapter memory pools */
    struct qdio_initialize    qdio_init_data;    /* for qdio_establish */
    struct device              generic_services; /* directory for WKA ports */
+   struct statistic_interface    *stat_if;
+   struct statistic              *stat_qdio_outb_full;
+   struct statistic              *stat_qdio_outb;
+   struct statistic              *stat_qdio_inb;
+   struct statistic              *stat_low_mem_scsi;
+   struct statistic              *stat_erp;
};

/*
@@ -1024,6 +1032,24 @@ struct zfcpl_unit {
    struct scsi_device    *device;          /* scsi device struct pointer */
    struct zfcpl_erp_action    erp_action;    /* pending error recovery */
    atomic_t              erp_counter;
+   atomic_t              read_num;
+   atomic_t              write_num;
+   struct statistic_interface    *stat_if;
+   struct statistic              *stat_sizes_scsi_write;
+   struct statistic              *stat_sizes_scsi_read;
+   struct statistic              *stat_sizes_scsi_nodata;
+   struct statistic              *stat_sizes_scsi_nofit;
+   struct statistic              *stat_sizes_scsi_nomem;
+   struct statistic              *stat_sizes_timedout_write;
+   struct statistic              *stat_sizes_timedout_read;
+   struct statistic              *stat_sizes_timedout_nodata;
+   struct statistic              *stat_latencies_scsi_write;
+   struct statistic              *stat_latencies_scsi_read;
+   struct statistic              *stat_latencies_scsi_nodata;
+   struct statistic              *stat_pending_scsi_write;
+   struct statistic              *stat_pending_scsi_read;
+   struct statistic              *stat_erp;
+   struct statistic              *stat_erp_reset;
};

/* FSF request */
@@ -1050,7 +1076,8 @@ struct zfcpl_fsf_req {
    mempool_t              *pool;          /* used if request was allocated
                                         from emergency pool */
-   unsigned long long    issued;          /* request sent time (STCK) */
+   struct zfcpl_unit    *unit;
+   unsigned long long    received;
+   struct zfcpl_unit    *unit;
};
```

[patch 6/6] statistics infrastructure – exploitation: zfc

```
typedef void zfc_fsf_req_handler_t(struct zfc_fsf_req*);
diff -Nurp f/drivers/s390/scsi/zfc_erp.c g/drivers/s390/scsi/zfc_erp.c
--- f/drivers/s390/scsi/zfc_erp.c      2005-12-14 12:51:26.000000000 +0100
+++ g/drivers/s390/scsi/zfc_erp.c      2005-12-14 16:01:55.000000000 +0100
@@ -1624,10 +1624,12 @@ zfc_erp_strategy_check_unit(struct zfc
     switch (result) {
     case ZFCP_ERP_SUCCEEDED :
         atomic_set(&unit->erp_counter, 0);
+       statistic_inc(unit->stat_erp, 1);
         zfc_erp_unit_unblock(unit);
         break;
     case ZFCP_ERP_FAILED :
         atomic_inc(&unit->erp_counter);
+       statistic_inc(unit->stat_erp, -1);
         if (atomic_read(&unit->erp_counter) > ZFCP_MAX_ERPS)
             zfc_erp_unit_failed(unit);
         break;
@@ -1695,10 +1697,12 @@ zfc_erp_strategy_check_adapter(struct z
     switch (result) {
     case ZFCP_ERP_SUCCEEDED :
         atomic_set(&adapter->erp_counter, 0);
+       statistic_inc(adapter->stat_erp, 1);
         zfc_erp_adapter_unblock(adapter);
         break;
     case ZFCP_ERP_FAILED :
         atomic_inc(&adapter->erp_counter);
+       statistic_inc(adapter->stat_erp, -1);
         if (atomic_read(&adapter->erp_counter) > ZFCP_MAX_ERPS)
             zfc_erp_adapter_failed(adapter);
         break;
diff -Nurp f/drivers/s390/scsi/zfc_ext.h g/drivers/s390/scsi/zfc_ext.h
--- f/drivers/s390/scsi/zfc_ext.h      2005-10-28 02:02:08.000000000 +0200
+++ g/drivers/s390/scsi/zfc_ext.h      2005-12-14 16:01:55.000000000 +0100
@@ -203,4 +203,10 @@ extern void zfc_scsi_dbf_event_abort(co
extern void zfc_scsi_dbf_event_devreset(const char *, u8, struct zfc_unit *,
                                         struct scsi_cmnd *);

+/**
+ * ***** stat *****
+ */
+extern int zfc_adapter_statistic_register(struct zfc_adapter *);
+extern int zfc_adapter_statistic_unregister(struct zfc_adapter *);
+extern int zfc_unit_statistic_register(struct zfc_unit *);
+extern int zfc_unit_statistic_unregister(struct zfc_unit *);
+
#endif /* ZFCP_EXT_H */
diff -Nurp f/drivers/s390/scsi/zfc_fsf.c g/drivers/s390/scsi/zfc_fsf.c
--- f/drivers/s390/scsi/zfc_fsf.c      2005-12-14 12:51:26.000000000 +0100
+++ g/drivers/s390/scsi/zfc_fsf.c      2005-12-14 16:01:55.000000000 +0100
@@ -219,6 +219,8 @@ zfc_fsf_req_complete(struct zfc_fsf_re
     int retval = 0;
     int cleanup;

+     fsf_req->received = get_clock();
+     if (unlikely(fsf_req->fsf_command == FSF_QTCB_UNSOLICITED_STATUS)) {
```

[patch 6/6] statistics infrastructure – exploitation: zfcpl

```

        ZFCP_LOG_DEBUG("Status read response received\n");
        /*
@@ -3471,6 +3473,12 @@ zfcpl_fsf_send_fcp_command_task(struct zf
                unit->fcp_lun,
                unit->port->wwpn,
                zfcpl_get_busid_by_adapter(adapter));
+
        if (retval == -ENOMEM)
+
                statistic_inc(unit->stat_sizes_scsi_nomem,
+
                                scsi_cmnd->request_bufflen);
+
        if (retval == -EIO)
+
                statistic_inc(unit->stat_sizes_scsi_nofit,
+
                                scsi_cmnd->request_bufflen);
+
        goto failed_req_create;
    }

@@ -3581,6 +3589,8 @@ zfcpl_fsf_send_fcp_command_task(struct zf
    zfcpl_erp_unit_shutdown(unit, 0);
    retval = -EINVAL;
    }
+
    statistic_inc(unit->stat_sizes_scsi_nofit,
+
                    scsi_cmnd->request_bufflen);
+
    goto no_fit;
    }

@@ -3591,6 +3601,13 @@ zfcpl_fsf_send_fcp_command_task(struct zf
    ZFCP_HEX_DUMP(ZFCP_LOG_LEVEL_DEBUG,
                  (char *) scsi_cmnd->cmnd, scsi_cmnd->cmd_len);

+
    if (scsi_cmnd->sc_data_direction == DMA_FROM_DEVICE)
+
        statistic_inc(unit->stat_pending_scsi_read,
+
                        atomic_inc_return(&unit->read_num));
+
    else if (scsi_cmnd->sc_data_direction == DMA_TO_DEVICE)
+
        statistic_inc(unit->stat_pending_scsi_write,
+
                        atomic_inc_return(&unit->write_num));
+
    /*
     * start QDIO request for this FSF request
     * covered by an SBALE)
@@ -3613,6 +3630,11 @@ zfcpl_fsf_send_fcp_command_task(struct zf
    goto success;

send_failed:
+
    if (scsi_cmnd->sc_data_direction == DMA_FROM_DEVICE)
+
        atomic_dec(&unit->read_num);
+
    else if (scsi_cmnd->sc_data_direction == DMA_TO_DEVICE)
+
        atomic_dec(&unit->write_num);
+
no_fit:
failed_scsi_cmnd:
    zfcpl_unit_put(unit);
@@ -3984,9 +4006,32 @@ zfcpl_fsf_send_fcp_command_task_handler(s
    u32 sns_len;

```

[patch 6/6] statistics infrastructure – exploitation: zfcpl

```

char *fcp_rsp_info = zfcpl_get_fcp_rsp_info_ptr(fcp_rsp_iu);
unsigned long flags;
+ struct zfcpl_adapter *adapter = fsf_req->adapter;
+ struct zfcpl_unit *unit = fsf_req->unit;
+ long long unsigned latency;

- read_lock_irqsave(&fsf_req->adapter->abort_lock, flags);
+ statistic_lock(unit->stat_if, flags);
+ latency = fsf_req->received - fsf_req->issued;
+ do_div(latency, 1000000);
+ latency++;
+ if (fcp_cmnd_iu->wddata == 1) {
+     statistic_inc_nolock(unit->stat_sizes_scsi_write,
+                          zfcpl_get_fcp_dl(fcp_cmnd_iu));
+     statistic_inc_nolock(unit->stat_latencies_scsi_write, latency);
+     atomic_dec(&unit->write_num);
+ } else if (fcp_cmnd_iu->rddata == 1) {
+     statistic_inc_nolock(unit->stat_sizes_scsi_read,
+                          zfcpl_get_fcp_dl(fcp_cmnd_iu));
+     statistic_inc_nolock(unit->stat_latencies_scsi_read, latency);
+     atomic_dec(&unit->read_num);
+ } else {
+     statistic_inc_nolock(unit->stat_sizes_scsi_nodata,
+                          zfcpl_get_fcp_dl(fcp_cmnd_iu));
+     statistic_inc_nolock(unit->stat_latencies_scsi_nodata, latency);
+ }
+ statistic_unlock(unit->stat_if, flags);
+
+ read_lock_irqsave(&adapter->abort_lock, flags);
+ scpnt = (struct scsi_cmnd *) fsf_req->data;
+ if (unlikely(!scpnt)) {
+     ZFCPL_LOG_DEBUG
@@ -4188,7 +4233,7 @@ zfcpl_fsf_send_fcp_command_task_handler(s
+     * Note: scsi_done must not block!
+     */
+ out:
- read_unlock_irqrestore(&fsf_req->adapter->abort_lock, flags);
+ read_unlock_irqrestore(&adapter->abort_lock, flags);
+ return retval;
+ }

@@ -4605,10 +4650,14 @@ zfcpl_fsf_req_sbal_get(struct zfcpl_adapte
+                                     ZFCPL_SBAL_TIMEOUT);
+     if (ret < 0)
+         return ret;
-     if (!ret)
+     if (!ret) {
+         statistic_inc(adapter->stat_qdio_outb_full, 1);
+         return -EIO;
-     } else if (!zfcpl_fsf_req_sbal_check(lock_flags, req_queue, 1))
+     }
+     } else if (!zfcpl_fsf_req_sbal_check(lock_flags, req_queue, 1)) {
+         statistic_inc(adapter->stat_qdio_outb_full, 1);
+         return -EIO;
+     }
+ }

```

[patch 6/6] statistics infrastructure – exploitation: zfc

```

        return 0;
    }
@@ -4774,6 +4823,9 @@ zfc_fsf_req_send(struct zfc_fsf_req *f
    * position of first one
    */
    atomic_sub(fsf_req->sbal_number, &req_queue->free_count);
+   statistic_inc(adapter->stat_qdio_outb,
+               QDIO_MAX_BUFFERS_PER_Q -
+               atomic_read(&req_queue->free_count));
    ZFCP_LOG_TRACE("free_count=%d\n", atomic_read(&req_queue->free_count));
    req_queue->free_index += fsf_req->sbal_number; /* increase */
    req_queue->free_index %= QDIO_MAX_BUFFERS_PER_Q; /* wrap if needed */
diff -Nurp f/drivers/s390/scsi/zfc_qdio.c g/drivers/s390/scsi/zfc_qdio.c
--- f/drivers/s390/scsi/zfc_qdio.c      2005-10-28 02:02:08.000000000 +0200
+++ g/drivers/s390/scsi/zfc_qdio.c      2005-12-14 16:01:55.000000000 +0100
@@ -418,6 +418,7 @@ zfc_qdio_response_handler(struct ccw_de
    } else {
        queue->free_index += count;
        queue->free_index %= QDIO_MAX_BUFFERS_PER_Q;
+       statistic_inc(adapter->stat_qdio_inb, count);
        atomic_set(&queue->free_count, 0);
        ZFCP_LOG_TRACE("%i buffers enqueued to response "
@@ -662,6 +663,10 @@ zfc_qdio_sbals_from_segment(struct zfc
    /* get next free SBALE for new piece */
    if (NULL == zfc_qdio_sbale_next(fsf_req, sbtype)) {
        /* no SBALE left, clean up and leave */
+       statistic_inc(
+           fsf_req->adapter->stat_qdio_outb_full,
+           atomic_read(
+               &fsf_req->adapter->request_queue.free_count));
        zfc_qdio_sbals_wipe(fsf_req);
        return -EINVAL;
    }
diff -Nurp f/drivers/s390/scsi/zfc_scsi.c g/drivers/s390/scsi/zfc_scsi.c
--- f/drivers/s390/scsi/zfc_scsi.c      2005-12-14 12:51:26.000000000 +0100
+++ g/drivers/s390/scsi/zfc_scsi.c      2005-12-14 16:01:55.000000000 +0100
@@ -449,6 +449,16 @@ zfc_scsi_ah_abort_handler(struct scsi_c
    ZFCP_LOG_INFO("aborting scsi_cmnd=%p on adapter %s\n",
                  scpnt, zfc_get_busid_by_adapter(adapter));

+   if (scpnt->sc_data_direction == DMA_TO_DEVICE)
+       statistic_inc(unit->stat_sizes_timedout_write,
+                   scpnt->request_bufflen);
+   else if (scpnt->sc_data_direction == DMA_FROM_DEVICE)
+       statistic_inc(unit->stat_sizes_timedout_read,
+                   scpnt->request_bufflen);
+   else
+       statistic_inc(unit->stat_sizes_timedout_nodata,
+                   scpnt->request_bufflen);

    /* avoid race condition between late normal completion and abort */
    write_lock_irqsave(&adapter->abort_lock, flags);

```

[patch 6/6] statistics infrastructure – exploitation: zfcps

```

@@ -538,12 +548,14 @@ zfcps_scsi_eh_device_reset_handler(struct
                                atomic_set_mask
                                (ZFCP_STATUS_UNIT_NOTSUPPUNITRESET,
                                 &unit->status);
+
                                statistic_inc(unit->stat_eh_reset, -1);
                                /* fall through and try 'target reset' next */
                                } else {
                                ZFCP_LOG_DEBUG("unit reset succeeded (unit=%p)\n",
                                                unit);
                                /* avoid 'target reset' */
                                retval = SUCCESS;
+
                                statistic_inc(unit->stat_eh_reset, 1);
                                goto out;
                                }
                                }
@@ -551,9 +563,11 @@ zfcps_scsi_eh_device_reset_handler(struct
if (retval) {
    ZFCP_LOG_DEBUG("target reset failed (unit=%p)\n", unit);
    retval = FAILED;
+
    statistic_inc(unit->stat_eh_reset, -2);
} else {
    ZFCP_LOG_DEBUG("target reset succeeded (unit=%p)\n", unit);
    retval = SUCCESS;
+
    statistic_inc(unit->stat_eh_reset, 2);
}
out:
    return retval;
diff -Nurp f/drivers/s390/scsi/zfcps_stat.c g/drivers/s390/scsi/zfcps_stat.c
--- f/drivers/s390/scsi/zfcps_stat.c      1970-01-01 01:00:00.000000000 +0100
+++ g/drivers/s390/scsi/zfcps_stat.c      2005-12-14 16:01:55.000000000 +0100
@@ -0,0 +1,195 @@
+/*
+ *
+ * linux/drivers/s390/scsi/zfcps_stat.c
+ *
+ * FCP adapter driver for IBM eServer zSeries
+ *
+ * Statistics
+ *
+ * (C) Copyright IBM Corp. 2005
+ *
+ * This program is free software; you can redistribute it and/or modify
+ * it under the terms of the GNU General Public License as published by
+ * the Free Software Foundation; either version 2, or (at your option)
+ * any later version.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
+ * GNU General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software
+ * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+ */
+
+#define ZFCP_STAT_REVISION "$Revision: 1.9 $"
+
+#include <linux/statistic.h>
+#include <linux/ctype.h>
+#include "zfcps_ext.h"
+

```

[patch 6/6] statistics infrastructure – exploitation: zfcpl

```
+#define ZFCP_LOG_AREA                ZFCP_LOG_AREA_OTHER
+
+int zfcpl_adapter_statistic_register(struct zfcpl_adapter *adapter)
+{
+    int retval = 0;
+    char name[14];
+
+    sprintf(name, "zfcpl-%s", zfcpl_get_busid_by_adapter(adapter));
+    statistic_interface_create(&adapter->stat_if, name);
+
+    retval |=
+        statistic_create(&adapter->stat_qdio_outb_full, adapter->stat_if,
+            "occurrence_qdio_outb_full",
+            "sbals_left/incidents");
+    statistic_define_value(adapter->stat_qdio_outb_full,
+        STATISTIC_RANGE_MIN, STATISTIC_RANGE_MAX,
+        STATISTIC_DEF_MODE_INC);
+
+    retval |= statistic_create(&adapter->stat_qdio_outb, adapter->stat_if,
+        "util_qdio_outb",
+        "slots-occupied/incidents");
+    statistic_define_range(adapter->stat_qdio_outb,
+        0, QDIO_MAX_BUFFERS_PER_Q);
+
+    retval |= statistic_create(&adapter->stat_qdio_inb, adapter->stat_if,
+        "util_qdio_inb", "slots-occupied/incidents");
+    statistic_define_range(adapter->stat_qdio_inb,
+        0, QDIO_MAX_BUFFERS_PER_Q);
+
+    retval |=
+        statistic_create(&adapter->stat_low_mem_scsi, adapter->stat_if,
+            "occurrence_low_mem_scsi", "-/incidents");
+    statistic_define_value(adapter->stat_low_mem_scsi, STATISTIC_RANGE_MIN,
+        STATISTIC_RANGE_MAX, STATISTIC_DEF_MODE_INC);
+
+    retval |= statistic_create(&adapter->stat_erp, adapter->stat_if,
+        "occurrence_erp", "results/incidents");
+    statistic_define_value(adapter->stat_erp,
+        STATISTIC_RANGE_MIN, STATISTIC_RANGE_MAX,
+        STATISTIC_DEF_MODE_INC);
+
+    return retval;
+}
+
+int zfcpl_adapter_statistic_unregister(struct zfcpl_adapter *adapter)
+{
+    return statistic_interface_remove(&adapter->stat_if);
+}
+
+int zfcpl_unit_statistic_register(struct zfcpl_unit *unit)
+{
+    int retval = 0;
+    char name[64];
+
+    atomic_set(&unit->read_num, 0);
+    atomic_set(&unit->write_num, 0);
+
+    sprintf(name, "zfcpl-%s-0x%016Lx-0x%016Lx",
+        zfcpl_get_busid_by_unit(unit), unit->port->wwpn, unit->fcpl_lun);
+    statistic_interface_create(&unit->stat_if, name);
+
+    retval |= statistic_create(&unit->stat_sizes_scsi_write, unit->stat_if,
```

[patch 6/6] statistics infrastructure – exploitation: zfcpl

```
+         "request_sizes_scsi_write",
+         "bytes/incidents");
+ statistic_define_list(unit->stat_sizes_scsi_write, 0,
+         STATISTIC_RANGE_MAX, 256);
+
+ retval |= statistic_create(&unit->stat_sizes_scsi_read, unit->stat_if,
+         "request_sizes_scsi_read",
+         "bytes/incidents");
+ statistic_define_list(unit->stat_sizes_scsi_read, 0,
+         STATISTIC_RANGE_MAX, 256);
+
+ retval |= statistic_create(&unit->stat_sizes_scsi_nodata, unit->stat_if,
+         "request_sizes_scsi_nodata",
+         "bytes/incidents");
+ statistic_define_value(unit->stat_sizes_scsi_nodata,
+         STATISTIC_RANGE_MIN, STATISTIC_RANGE_MAX,
+         STATISTIC_DEF_MODE_INC);
+
+ retval |= statistic_create(&unit->stat_sizes_scsi_nofit, unit->stat_if,
+         "request_sizes_scsi_nofit",
+         "bytes/incidents");
+ statistic_define_list(unit->stat_sizes_scsi_nofit, 0,
+         STATISTIC_RANGE_MAX, 256);
+
+ retval |= statistic_create(&unit->stat_sizes_scsi_nomem, unit->stat_if,
+         "request_sizes_scsi_nomem",
+         "bytes/incidents");
+ statistic_define_value(unit->stat_sizes_scsi_nomem, STATISTIC_RANGE_MIN,
+         STATISTIC_RANGE_MAX, STATISTIC_DEF_MODE_INC);
+
+ retval |=
+     statistic_create(&unit->stat_sizes_timedout_write, unit->stat_if,
+         "request_sizes_timedout_write", "bytes/incidents");
+ statistic_define_value(unit->stat_sizes_timedout_write,
+         STATISTIC_RANGE_MIN, STATISTIC_RANGE_MAX,
+         STATISTIC_DEF_MODE_INC);
+
+ retval |=
+     statistic_create(&unit->stat_sizes_timedout_read, unit->stat_if,
+         "request_sizes_timedout_read", "bytes/incidents");
+ statistic_define_value(unit->stat_sizes_timedout_read,
+         STATISTIC_RANGE_MIN, STATISTIC_RANGE_MAX,
+         STATISTIC_DEF_MODE_INC);
+
+ retval |=
+     statistic_create(&unit->stat_sizes_timedout_nodata, unit->stat_if,
+         "request_sizes_timedout_nodata",
+         "bytes/incidents");
+ statistic_define_value(unit->stat_sizes_timedout_nodata,
+         STATISTIC_RANGE_MIN, STATISTIC_RANGE_MAX,
+         STATISTIC_DEF_MODE_INC);
+
+ retval |=
+     statistic_create(&unit->stat_latencies_scsi_write, unit->stat_if,
+         "latencies_scsi_write", "milliseconds/incidents");
+ statistic_define_array(unit->stat_latencies_scsi_write, 0, 1024, 1,
+         STATISTIC_DEF_SCALE_LOG2);
+
+ retval |=
+     statistic_create(&unit->stat_latencies_scsi_read, unit->stat_if,
+         "latencies_scsi_read", "milliseconds/incidents");
+ statistic_define_array(unit->stat_latencies_scsi_read, 0, 1024, 1,
```

[patch 6/6] statistics infrastructure – exploitation: zfc

```
+             STATISTIC_DEF_SCALE_LOG2);
+
+     retval |=
+         statistic_create(&unit->stat_latencies_scsi_nodata, unit->stat_if,
+             "latencies_scsi_nodata", "milliseconds/incidents");
+     statistic_define_array(unit->stat_latencies_scsi_nodata, 0, 1024, 1,
+         STATISTIC_DEF_SCALE_LOG2);
+
+     retval |=
+         statistic_create(&unit->stat_pending_scsi_write, unit->stat_if,
+             "pending_scsi_write", "commands/incidents");
+     statistic_define_range(unit->stat_pending_scsi_write, 0,
+         STATISTIC_RANGE_MAX);
+
+     retval |= statistic_create(&unit->stat_pending_scsi_read, unit->stat_if,
+         "pending_scsi_read", "commands/incidents");
+     statistic_define_range(unit->stat_pending_scsi_read, 0,
+         STATISTIC_RANGE_MAX);
+
+     retval |= statistic_create(&unit->stat_erp, unit->stat_if,
+         "occurrence_erp", "results/incidents");
+     statistic_define_value(unit->stat_erp,
+         STATISTIC_RANGE_MIN, STATISTIC_RANGE_MAX,
+         STATISTIC_DEF_MODE_INC);
+
+     retval |= statistic_create(&unit->stat_eh_reset, unit->stat_if,
+         "occurrence_eh_reset", "results/incidents");
+     statistic_define_value(unit->stat_eh_reset,
+         STATISTIC_RANGE_MIN, STATISTIC_RANGE_MAX,
+         STATISTIC_DEF_MODE_INC);
+
+     return retval;
+ }
+
+ int zfc_unit_statistic_unregister(struct zfc_unit *unit)
+ {
+     return statistic_interface_remove(&unit->stat_if);
+ }
+
+ #undef ZFCP_LOG_AREA
+
+ -
```

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxxxx
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>