

## OpenVMS- Sicherheitsschnittstellen für Execlets in C geschrieben

1G05: OpenVMS Sicherheitsschnittstellen für Execlets in C geschrieben

John R. Covert

DECUS-München    16. Mai 2006    Düsseldorf

### Execlets: Was, warum, wie?

- Programm und ggf. data
- Non-paged sections erlauben high-IPL
- Loadable. Unloadable, ggf. nicht-unloadable
- URKW
- Init routine deallocatable nach laden
- Unload routine nicht 100% standardisiert.
- 
- Wie schreibt man ein Execlet in C
- 
- Was sind einige interessante Sicherheitsschnittstellen

## Execlet Beispiel

- Initmodule und eine oder mehrere Codemodules

```
#pragma module execlet_init "X-1"
#pragma code_psect "EXEC$INIT_CODE"
int init(struct _ldring *ldring, int flags,
        struct user_buffer *user_buffer);
#define INIT000_ROUTINE init
#include <init_rtn_setup.c>
int init(struct _ldring *ldring, int flags,
        struct user_buffer *user_buffer) {
    /* Wird in Kernel Mode gerufen */
    /* ... */
    return(initializationok);
}
```

## Execlet Beispiel (2)

```
#pragma module execlet "X-1"
/*execlet.c */
extern PCB * const ctl$gl_pcb;
int unload(), routine2();
struct _sdexstatic {
    char unldsign[8]; /* Allow the Unloader to verify presence of
                       unload routine */
    int (*p_unload)(); /* Unload routine pointer */
    long datum1;
    int (*p_routine2)(); /* Pointer to some callable routine */
    struct global_data_struct *gds;
} globaldef {"$000LOW$"}
struct _execletstatic exvec
= { "UNLDRTN:", unload, 0 /*datum1*/, routine2 };
int unload() {
    /* ... */
    return(readytounload);
}
```



**Execlet laden**

```
#include <ldrdef.h>
#include <ldringdef.h>
static struct
    {unsigned int imgaddr; struct _ldring *ldringaddr; unsigned int
      seqno;}
      ref_handle={0,0,0};
$DESCRIPTOR(execlet_d, "DEV:[dir]execlet.exe");
load() {
    return ldr$load_image(&execlet_d, LDR$M_UNL|LDR$M_USER_BUF,
                          &ref_handle, &user_buffer);
}
status = sys$cmexec(load,0);
if (!$VMS_STATUS_SUCCESS(status)) return(status);
```

**Zugriff auf Daten und Routinen in PSECT \$000LOW\$**

IA64: ref\_handle.ldringaddr->ldring\$l\_segments->ldrisd\$p\_base

Alpha:ref\_handle.ldringaddr->ldring\$l\_segments->
 ldring\$l\_nonpag\_w\_base

## Execlet aufsuchen

```

void info_routine() {
extern struct _ldrimg *ldr$gq_image_list;
extern MUTEX exe$gq_basimgmtx;
sch_std$lockr_quad(&exe$gq_basimgmtx, ctl$gl_pcb); /* mutex while in this list */
struct _ldrimg *p = ldr$gq_image_list;
while (p != (struct _ldrimg *)&ldr$gq_image_list) {
    if (p->ldrimg$l_imgnamlen == execllet_d.dsc$w_length
        && memcmp(p->ldrimg$ps_imgnam, execllet_d.dsc$a_pointer,
            execllet_d.dsc$w_length) == 0) {
        ref_handle.ldrimgaddr = p;
        ref_handle.seqno = p->ldrimg$l_seq;
        ref_handle.imgaddr=(unsigned int)
#ifdef __ia64
            p->ldrimg$l_segments->ldrisd$p_base;
#else
            p->ldrimg$l_nonpag_w_base;
#endif
        break;
    }
    p = p->ldrimg$l_flink;
} /* if we don't break out, ref_handle.ldrimgaddr remains zero */
sch_std$unlock_quad(&exe$gq_basimgmtx, ctl$gl_pcb);
}
sys$cmkrnl(info_routine,0);
    
```

## Execlet in SDA

```

SDA> show exe pw*
Image                               Base                               End                               Length                               ImageOff                               SymVec
-----
PWIPDRIVER                           FFFFFFFF.82EDE000 FFFFFFFF.82EEDFFF 00000000.00010000
Nonpaged read only                   FFFFFFFF.82EE2000 FFFFFFFF.82EED1FF 00000000.00008200 00004000
Nonpaged read/write                  FFFFFFFF.82EDE000 FFFFFFFF.82EE11FF 00000000.00003200 00000000
Linked 21-NOV-2004 19:08             LDRIMG 815FFE80      SeqNum 0000009E --<not sliced >--

SDA> show exe f*
F11BXQP                               FFFFFFFF.801FA000 FFFFFFFF.8024D5FF 00000000.00053600 00000000
Nonpaged read only                   FFFFFFFF.81119200 FFFFFFFF.811263FF 00000000.00000200 00054000
Nonpaged read/write                  LDRIMG 814362C0      SeqNum 00000024 --<sliced >--
Linked 19-NOV-2004 08:58
    
```

## Execlet Device Dismount

```
$ show dev sys$sysdevice:/files
```

```
...
```

```
00000000 [VMS$COMMON.SYS$LDR]F11BXQP.EXE;1
```

- Execlet auf sysdevice, kein Problem
- Execlet woanders:

```
$ dismount dka100
```

```
%DISM-W-CANNOTDMT, DKA100 cannot be dismounted
```

```
%DISM-W-USERFILES, 1 user file open on volume
```

- Lösung

```
WCB *wcb = ldrimg->ldrimg$l_wcb; /* Check not zero */
```

```
ldrimg->ldrimg$l_wcb = 0;
```

```
wcb->wcb$l_refcnt = 0;
```

```
_INSQHI(&wcb->wcb$q_delinq, &exe$gg_wcbdelinq);
```

```
mmg_std$delgblwcb(ctl$gl_pcb);
```

## Interessante Schnittstellen

- Image Activation callout
- File Access callout
- Search Rights Array
- ACME Password Check
- Process Events

## Image Activation Callout

```

! Call any external components
begin
  LOCAL
    next;    ! pointer to next routine to call
  global
    EXE$GQ_IMGACT_SECURITY : alias;

    next = EXE$GQ_IMGACT_SECURITY;    ! begin at list head
    while ((next = ..next) neq 0)
      do (.(next+8))(.icb_adr);
end;

ERRCHK( END_PROCESSING (.ICB_ADR)); ! Set final state

```

## Image Activation Callout (2)

```

#include <imcbdef.h>
typedef struct _imgact_callout
{
  struct _imgact_callout * flink;
  int32 unused;
  void (*routine)(IMCB *imcb);
} IMGACT_CALLOUT;

/* Must be in system space */
static IMGACT_CALLOUT ourblock = {0,0,imgact_callout};
extern IMGACT_CALLOUT *EXE$GQ_IMGACT_SECURITY;
/* Use this Mutex */
sch_std$lockw_quad(&ioc$gq_mutex, ctl$gl_pcb);
/* Link at head */
ourblock.flink = EXE$GQ_IMGACT_SECURITY->flink;
EXE$GQ_IMGACT_SECURITY->flink = &ourblock;

sch_std$unlock_quad(&ioc$gq_mutex, ctl$gl_pcb);

```

## Image Activation Callout (3)

```

void imgact_callout( IMCB *imcb) {
  /* EXEC mode, process context */
  if (imcb->imcb$b_act_code == IMCB$b_K_MAIN_PROGRAM) {
    int64 prvprv;
    sys$setprv(1, &(PRV$b_M_CMEEXEC), 0, &prvprv);
    sys$dclast(imgact_callout_s,prvprv,PSL$b_C_SUPER);
  }
}

void imgact_callout_s(int64 prvprv) {
  /* Super mode, vor dem ersten User Instruction */
  struct {int64 argc; int64 p1;} arglst = { 1, prvprv};
  sys$cmexec_64(imgact_callout_e,&arglst);
}

void imgact_callout_e(int64 prvprv) {
  /* Get rid of CMEEXEC if he didn't have it */
  prvprv = (!prvprv) & (PRV$b_M_CMEEXEC);
  if (prvprv) sys$setprv(0, &prvprv, 0, 0);
  /* Und dann alles anderes machen */
}

```

## File Access Callout

```

BEGIN ! [F11X]CHKPRO before return from routine CHECK_PROTECT
EXTERN
  EXE$b_GQ_XQP_SECURITY : WEAK ALIAS, ! Future VMS version
  XQP$b_AR_XQP_PRIVATE : ALIAS ; ! If NEQ 0, remedial support
LOCAL
  ROUTINE_LIST : REF BBLOCK ;
MACRO
  CALLOUT_FLINK = 0, 0, 32, 1%,
  CALLOUT_ROUTINE = 8, 0, 32, 1% ;
IF ((ROUTINE_LIST = EXE$b_GQ_XQP_SECURITY) EQLA 0) THEN
  ROUTINE_LIST = XQP$b_AR_XQP_PRIVATE ;
WHILE ((ROUTINE_LIST = .ROUTINE_LIST[CALLOUT_FLINK]) LSS 0) DO
  IF (.ROUTINE_LIST[CALLOUT_ROUTINE] LSS 0) THEN
    (.ROUTINE_LIST[CALLOUT_ROUTINE])(.USER_PROFILE, .FCB,
    .CURRENT_VCB, .FILE_AUDIT[NSA_L_ACCESS], .STATUS,
    .ACMODE, .CHPRET);
END ;

```

## File Access Callout (2)

```
typedef struct _xqchkpro_callout
{
    struct _xqchkpro_callout * flink;
    int32 unused;
    void (*routine)(int64 profile, struct _fcb *fcb,
                   struct _vcb *vcb, int64 nsa_l_access,
                   int64 chkpro_status, int64 acmode, int64 chpret);
} XQPCHKPRO_CALLOUT;
/* System space */
static XQPCHKPRO_CALLOUT xqp_callout_block = {0,0,xqp_callout};
extern XQPCHKPRO_CALLOUT *EXE$GQ_XQP_SECURITY;

sch_std$lockw_quad(&ioc$gq_mutex, ctl$gl_pcb);
/* Note well that the EXE$GQ_XQP_SECURITY does not work exactly like
   EXE$GQ_IMGACT_SECURITY */
xqp_callout_block.flink = EXE$GQ_XQP_SECURITY;
/* link directly at EXE$GQ_XQP_SECURITY */
EXE$GQ_XQP_SECURITY = &xqp_callout_block;

sch_std$unlock_quad(&ioc$gq_mutex, ctl$gl_pcb);
```

## File Access Callout (3)

```
void xqp_callout(int64 profile, struct _fcb *fcb,
                 struct _vcb *vcb, int64 nsa_l_access,
                 int64 chkpro_status, int64 acmode, int64 chpret) {
/* Kernel mode, process or system context, XQP operation in progress
*/
}
```

## Search Rights Array

- Exec Routine

```
int exe_std$search_rights_array(unsigned int, struct _psb *, void *,
                               void *);
```

- Typical Use

```
/* search_rights_array must be called in EXEC or KERNEL mode in the
 * context of the process whose rights are to be searched
 */
```

```
int search_rights_array(unsigned int right) {
    long long dummy;
    return $VMS_STATUS_SUCCESS(exe_std$search_rights_array(right,
        cti$gl_pcb->pcb$ar_natural_psb, &dummy, &dummy));
}
```

## ACME password check

Muß nicht unbedingt in ein Execlet sein...

```
int status;
struct _acmesb acmsb;
int acmlogontype = ACME$K_LOCAL;
/* Rant: Jobs on FTAnn: devices (e.g. SSH and DECterm) do not have
 * jib$_jobtype filled in, which causes default processing in
 * ACME to return the CONTACTSYSMGR message. The ACME log will
 * contain a message that the call omitted ACME$LOGON_TYPE.
 * We work around that problem by specifying it as LOCAL to ACME,
 * and then in the bizarre case that we have a user who is not
 * allowed LOCAL access, we set ACME$M_NOAUTHORIZATION, since
 * we only need to authenticate. Using LOCAL isn't as bizarre as
 * it might seem, since even TELNET sessions are considered LOCAL anyway.
 * REMOTE only gets set for DECnet RTAN terms. If jib$_jobtype
 * starts being given a reasonable value post VMS V8.2, we can
 * remove this hack.
 */
struct {short int len, code; char *bufadr; int retlen;
        short int len2, code2; int *bufadr2; int retlen2;
        short int lenpw2, codepw2; char *bufadrpw2; int retlenpw2;
        int terminator;}
acmelist = {pwdcnt, ACME$PASSWORD_1, pwd, 0,
            sizeof(int), ACME$LOGON_TYPE, &acmlogontype, 0, 0, 0};
```

## ACME Password (2)

```

/* Haben wir nur ein oder zwei Passwörter? */
for (int i=0; i<(pwdcnt-1); i++) {
    if (pwd[i] == '/') {
        acmelist.len = i;
        acmelist.codepw2 = ACME$_PASSWORD_2;
        acmelist.bufadrpw2 = &term->pwd[i+1];
        acmelist.lenpw2 = (term->pwdcnt-i)-1;
        acmelist.retlenpw2 = 0;
        acmelist.terminator = 0;
        break;
    }
}
status = sys$acmw(EFNS$_ENF, /* no event flag */
    ACME$_FC_AUTHENTICATE_PRINCIPAL /* ACM function code */
    | ACME$_DEFAULT_PRINCIPAL
    | ACME$_NOAUTHORIZATION, /* See rant above */
    0, /* pointer to Context pointer */
    &acmelist, /* Item List */
    &acmsb, /* ACM Status block */
    0, /* AST routine */
    0); /* AST Parameter */
if ($VMS_STATUS_SUCCESS(status)) status = acmsb.acmesb$l_status;
/* Never leave a plaintext password in memory */
memset(pwd, 0, MAX_PWD_SIZE);

```

## Process Events

- Nur für non-unloadable Execlets

```

#include <prcevtdef.h>
exe$declare_prcevt_handler(PRCEVT$_USRUNDWN_EXEC,
    &process_event_handler);
exe$declare_prcevt_handler(PRCEVT$_DELPRC_RUNDWN,
    &process_event_handler);
void process_event_handler(int type) {
    if (type == PRCEVT$_DELPRC_RUNDWN) { /* Process Rundown */
    } else {
    }
}

```

- Es gibt eine ganze Menge von möglichen Events. Siehe .h, suche in [SYS.LIS]

## Fragen?

John R Covert  
2 Flagg Road  
Acton, MA 01720  
covert@covert.org

Unabhängiger Konsultant für VMS  
Kernelcodierung und Telephonie

(030) 8687 0 9350  
+1 978 263-5433

```

!          LINK_EXECLET.COM
!
$ if arch_name .eqs. "Alpha"
$ then
$ LINK /NATIVE_ONLY /BPAGE=14 /SECTION_BINDING -
      /REPLACE /NOTRACEBACK /SYSEXE=SELECTIVE /NOSYSSHR -
      /SHAREABLE=DEV_EXE:EXECLET - ! execlet image
      /SYMBOL=DEV_LIS:EXECLET -      ! Symbol table
      /MAP=DEV_LIS:EXECLET /FULL /CROSS - ! Map listing
      SYS$INPUT:/OPTIONS
!
!   Define symbol table for SDA using all global symbols, not just
!   universal ones
!
SYMBOL_TABLE=GLOBALS      ! Create symbol table for SDA.
CLUSTER=EXECLET,,,-
      SYS$LIBRARY:STARLET/INCLUDE:(SYS$DOINIT), -
      !
      !   Start with the execlet module
      !
      DEV_OBJ:EXECLET,EXECLET_INIT, -
      EXECLET_WEITERE_MODULE
      !
      SYS$LIBRARY:VMS$VOLATILE_PRIVATE_INTERFACES -
      /INCLUDE=(BUGCHECK_CODES)/LIB
!
PSECT_ATTR=$CODE$,PIC,USR,CON,REL,GBL,NOSHR,EXE,RD,NOWRT,NOVEC,MOD
PSECT_ATTR=_LIB$CODE$,PIC,USR,CON,REL,GBL,NOSHR,EXE,RD,NOWRT,NOVEC,MOD
!
PSECT_ATTR=$LINK$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,MOD
PSECT_ATTR=$LINKAGE$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,MOD
PSECT_ATTR=$LITERAL$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,MOD
PSECT_ATTR=$PLIT$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,MOD
PSECT_ATTR=$READONLY$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,MO
D
PSECT_ATTR=$READONLY_ADDR$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOV
EC,MOD
!
PSECT_ATTR=$000LOW$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,MOD
PSECT_ATTR=$OWN$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,MOD
PSECT_ATTR=$BSS$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,MOD
PSECT_ATTR=$DATA$,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,MOD
PSECT_ATTR=EXEC$UNL_000,PIC,USR,CON,REL,GBL,NOSHR,NOEXE,RD,WRT,NOVEC,
MOD
!
PSECT_ATTR=EXEC$INIT_LINKAGE,PIC,USR,CON,REL,GBL,NOSHR,EXE,RD,WRT,NOV
EC
!
COLLECT=NONPAGED_READWRITE_PSECTS/ATTRIBUTES=RESIDENT,-
      $000LOW$,-
      $LINK$,-
      $LINKAGE,-
      $LITERAL$,-
      $PLIT$,-
      $READONLY$,-
      $READONLY_ADDR$,-
!
      $OWN$,-
      $BSS$,-
      $DATA$,-
      EXEC$UNL_000
COLLECT=NONPAGED_READONLY_PSECTS/ATTRIBUTES=RESIDENT,-

```

```

        $CODE$, -
        _LIB$CODE
COLLECT=INITIALIZATION_PSECTS/ATTRIBUTES=INITIALIZATION_CODE, -
        EXEC$INIT_CODE, -
        EXEC$INIT_LINKAGE, -
        EXEC$INIT_000, -
        EXEC$INIT_001, -
        EXEC$INIT_002, -
        EXEC$INIT_PFNTBL_000, -
        EXEC$INIT_PFNTBL_001, -
        EXEC$INIT_PFNTBL_002, -
        EXEC$INIT_SSTBL_000, -
        EXEC$INIT_SSTBL_001, -
        EXEC$INIT_SSTBL_002
$ exit
$ endif
$ if arch_name .eqs. "IA64"
$ then
$ LINK /NATIVE_ONLY /BPAGE=14 -
        /NOTRACEBACK /SYSEXE=SELECTIVE /NOSYSSHR -
        /SHAREABLE=DEV_EXE:EXECLET - ! execlet image
        /SYMBOL=DEV_LIS:EXECLET - ! Symbol table
        /MAP=DEV_LIS:EXECLET /FULL /CROSS - ! Map listing
        SYS$INPUT:/OPTIONS
!
! Define symbol table for SDA using all global symbols, not just
! universal ones
!
SYMBOL_TABLE=GLOBALS ! Create symbol table for SDA.
CLUSTER=TEST_EXECLET, , , -
        SYS$LIBRARY:STARLET/INCLUDE:(SYS$DOINIT), -
        !
        ! Start with the execlet module
        !
        DEV_OBJ:EXECLET, EXECLET_INIT, -
        EXECLET_WEITERE_MODULES
        !
        SYS$LIBRARY:VMS$VOLATILE_PRIVATE_INTERFACES -
        /INCLUDE=(BUGCHECK_CODES)/LIB
!
PSECT_ATTR=$CODE$, CON, REL, GBL, NOSHR, EXE, RD, NOWRT, NOVEC, MOD
PSECT_ATTR=_LIB$CODE, CON, REL, GBL, NOSHR, EXE, RD, NOWRT, NOVEC, MOD
!
PSECT_ATTR=$LINK$, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
PSECT_ATTR=$LINKAGE, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
PSECT_ATTR=$LITERAL$, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
PSECT_ATTR=$PLIT$, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
PSECT_ATTR=$READONLY$, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
PSECT_ATTR=$READONLY_ADDR$, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
!
PSECT_ATTR=$000LOW$, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
PSECT_ATTR=$OWN$, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
PSECT_ATTR=$BSS$, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
PSECT_ATTR=$DATA$, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
PSECT_ATTR=EXEC$UNL_000, CON, REL, GBL, NOSHR, NOEXE, RD, WRT, NOVEC, MOD
!
PSECT_ATTR=EXEC$INIT_LINKAGE, CON, REL, GBL, NOSHR, EXE, RD, WRT, NOVEC
!
COLLECT=NONPAGED_READWRITE_PSECTS/ATTRIBUTES=RESIDENT, -
        $000LOW$, -
        $LINK$, -
        $LINKAGE, -

```

```
    $LITERAL$,-
    $SPLIT$,-
    $READONLY$,-
    $READONLY_ADDR$,-
!
    $OWN$,-
    $BSS$,-
    $DATA$,-
    EXEC$UNL_000
COLLECT=NONPAGED_READONLY_PSECTS/ATTRIBUTES=RESIDENT,-
    $CODE$,-
    _LIB$CODE
COLLECT=INITIALIZATION_PSECTS/ATTRIBUTES=INITIALIZATION_CODE,-
    EXEC$INIT_CODE,-
    EXEC$INIT_LINKAGE,-
    EXEC$INIT_000,-
    EXEC$INIT_001,-
    EXEC$INIT_002,-
    EXEC$INIT_PFNTBL_000,-
    EXEC$INIT_PFNTBL_001,-
    EXEC$INIT_PFNTBL_002,-
    EXEC$INIT_SSTBL_000,-
    EXEC$INIT_SSTBL_001,-
    EXEC$INIT_SSTBL_002
$ exit
$ endif
$ write sys$output "This procedure does not support architecture
",arch_name
$ exit
```