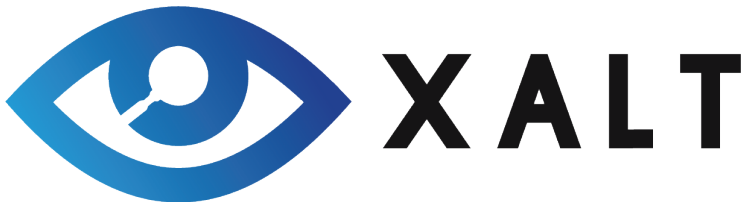


# Protecting XALT from Users

Robert McLay

March 17, 2022

# XALT: Outline



- ▶ XALT is linking with every program that runs on the system
- ▶ Users make mistakes
- ▶ Need to protect XALT from user mistakes
- ▶ Show three protection examples

# Three examples of protection

- ▶ User's Expecting allocated memory to be zero'd
- ▶ User's mixing Fortran routine with C library routines badly
- ▶ XALT expecting well managed memory heap.

# How XALT works

```
#include <stdio.h>
void myinit(int argc, char **argv)
{ printf("This is run before main()\n"); }
void myfini()
{ printf("This is run after main()\n"); }

__attribute__((section(".init_array"))) __typeof__(myinit) *__init = myinit;
__attribute__((section(".fini_array"))) __typeof__(myfini) *__fini = myfini;
```

- [my\\_docs/22/xalt\\_monthly\\_mtg\\_2022\\_03\\_17/code/bad\\_memory/ex1](#)

# How XALT works (II)

```
#include <stdio.h>
int main()
{
    printf("Hello World!\n");
    return 0;
}
```

# How XALT works (III)

```
$ ./try
```

```
Hello World!
```

```
$ LD_PRELOAD=./libxalt.so ./try
```

```
This is run before main()
```

```
Hello World!
```

```
This is run after main()
```

- ▶ `my_docs/22/xalt_monthly_mtg_2022_03_17/code/bad_memory/ex1`

# User's expecting memory to be zero'd when malloc()

- ▶ Initially all memory is zero'd before program starts
- ▶ Note that pointer zero, integer zero and float zero are all zeros
- ▶ Link lists require a NULL pointer at end of list.
- ▶ Used memory is **NOT** zero'd for you in C.
- ▶ User programmed work w/o XALT, Failed with XALT.

# Example code clean/used memory

% cat try.c

```
#include <stdio.h>
#include <stdlib.h>
#define SZ 1000
int main()
{
    int *a = (int *) malloc(SZ*sizeof(int));
    printf("Hello World! a:%d\n",a[0]);
    return 0;
}
```

% cat xalt.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SZ 1000
void myinit(int argc, char **argv)
{
    int i;
    int *a = (int*) malloc(SZ*sizeof(int));
    for (i = 0; i < SZ; ++i) a[i] = 15;
    free(a);
    printf("This is run before main()\n");
}
__attribute__((section(".init_array"))) __typeof__(myinit) *__init = myinit;
```

► [my\\_docs/22/xalt\\_monthly\\_mtg\\_2022\\_03\\_17/code/bad\\_memory/ex2](#)



# Example code clean/used memory(II)

```
% ./try
```

```
Hello World!  a:0
```

```
% LD_PRELOAD=./libxalt.so  ./try  ; echo
```

```
This is run before main()
```

```
Hello World!  a:15
```

```
This is run after main()
```

- ▶ my\_docs/22/xalt\_monthly\_mtg\_2022\_03\_17/code/bad\_memory/ex2

# XALT Fix: zero memory before free()

- ▶ To protect XALT from broken user code
- ▶ XALT in myinit() zero's memory before free
- ▶ Note that non-MPI tracking does little allocation
- ▶ MPI tasks  $> 127$  init record  $\Rightarrow$  much allocation

# XALT Fix: zero memory before free()

% cat try.c

```
#include <stdio.h>
#include <stdlib.h>
#define SZ 1000
int main()
{
    int *a = (int *) malloc(SZ*sizeof(int));
    printf("Hello World! a:%d\n",a[0]);
    return 0;
}
```

% cat xalt.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SZ 1000
void myinit(int argc, char **argv)
{
    int i;
    int *a = (int*) malloc(SZ*sizeof(int));
    for (i = 0; i < SZ; ++i) a[i] = 15;
    memset((void *) a, 0, SZ*sizeof(int));
    free(a);
    printf("This is run before main()\n");
}
__attribute__((section(".init_array"))) __typeof__(myinit) *__init = myinit;
```

► my\_docs/22/xalt\_monthly\_mtg\_2022\_03\_17/code/bad\_mem-

# XALT Fix: zero memory before free() (II)

```
% ./try
```

```
Hello World!  a:0
```

```
% LD_PRELOAD=./libxalt.so  ./try  ; echo
```

```
This is run before main()
```

```
Hello World!  a:0
```

```
This is run after main()
```

- ▶ my\_docs/22/xalt\_monthly\_mtg\_2022\_03\_17/code/bad\_memory/ex3

# Protecting XALT from Fortran mixed with C programs badly

```
% cat msg.f90
subroutine msg
  print *, "Hello World!"
end subroutine msg
```

```
% nm try | grep msg
000000000000011c7 T msg_
```

- ▶ Normally fortran routines get a trailing underscore when compiled
- ▶ gfortran: -fno-underscoring
- ▶ ifort: -assume nounderscore
- ▶ Will remove the trailing underscore
- ▶ Can make mixing C/Fortran easier
- ▶ Also make collisions with C library easier

# XALT uses libuuid

- ▶ libuuid.so is used to get a unique identifier
- ▶ It uses libc's random()
- ▶ You only get one name per program
- ▶ Can't have two routines named random()

# Collision over random() routine

```
% cat try.f90
```

```
program tryMe
  implicit none
  real*8 d, random
  print *, "Hello World!"
  d = random(1.0, 2.0, 3.0)
  print *, "d: ",d
end program tryMe
```

```
% cat random.f90
```

```
real*8 function random(a, b, c)
  implicit none
  real*8 a, b, c
  print *, "In random(a, b, c)"
  random = a*b + c
end function random
```

```
% cat xalt.c
```

```
#include <stdio.h>
#include <stdlib.h>
void myinit(int argc, char **argv)
{
  long int a;
  printf("This is run before main()\n");
  a = random();
  printf("called random(): a: %ld\n",a);
}
__attribute__((section(".init_array"))) __typeof__(myinit) *__init = myinit;
```

# Conclusions

- ▶ We have a way to track imports from R and Python
- ▶ It works well but there are a few conflicts with sitecustomize.py
- ▶ We have yet more data to try to figure out what to do with.



# Future Topics?

► Others?