Secure Operating System Design and Implementation Userspace

Jon A. Solworth

Dept. of Computer Science University of Illinois at Chicago

March 30, 2015

(ロ) (固) (目) (目) (目) (り)

Jon A. Solworth

Secure OS Design and Implementation

Userspa

Part I Userspace overview Jon A. Solworth Secure OS Design and Implementation Userspace Userspace Userspace Userspace

Userspace processes

In these slides we describe

- How Ethos C code differs from Linux C code
- How to compile userspace programs
- The initial userspace library
- The Ethos file system layout
- Test code
- We assume you know make, the Unix utility

Ethos tool chain

Xen the virtual machine monitor

subversion source control containing Ethos

/home/svn/projects/ethos/ethos the kernel

/home/svn/projects/ethos/ethosUserspace

userspace

gcc the gcc compiler suite

make build software

bugzilla report and track bugs



Overview

- Ethos is designed and implemented to change userspace programming
- Making it simpler, more reliable, and more secure
- And thus Ethos's interfaces are incompatibility with existing systems
- Its incompatible syscalls result in incompatible libraries
- Incompatible libraries result in incompatible programs
- Whew, there is a lot to build here
- Especially since libraries need improvement too

4 □ ト 4 □ ト 4 □ ト 4 □ ト 9 0 (

Jon A. Solworth

Secure OS Design and Implementation

Userspac

Primary goal for C-level libraries

- We don't believe in building substantial libraries in C
- Because they are more prone to security holes than higher level libraries
- Our primary reason for doing so is to enable porting of programming languages
- And so we're doing more library porting than we would like



Jon A. Solwort

Secure OS Design and Implementat

Heerenace

Part II

Creating userspace code

Building a process

- The program is written in some Programming Language (PL)
- It is compiled into an object file (typically in ELF format)
- The object file is linked to some libraries

static libraries ensure that the executable completely contains the userspace code

dynamic libraries produces smaller executables and map in libraries when the executable is loaded.

- Each PL a **standard library**
- Each PL implementation requires that its standard library be implemented
- (But it is possible to change the standard library, while keeping the PL).



Standard libraries

- Standard libraries include many OS syscalls and abstractions based on traditional OS semantics
- Higher level libraries are going to rely on the functionality of standard libraries
- Ethos's goal is to build simpler libraries which are easier to use and specialized to Ethos
- The most basic Ethos library is estdlib, containing procedure wrappers for syscalls
- We'll start to build on top of that
- (later on a higher level programming language Python.)

4□▶ 4□▶ 4□▶ 4□▶ □ 900

Jon A. Solworth

Secure OS Design and Implementation

Userspac

Part III

Compiling userspace programs



Jon A. Solwort

Secure OS Design and Implementation

Userspac

Compilation

- Ethos userspace programs, as with the Ethos kernel, is compiled under Dom0.
- Dom0 is Linux
- Ethos compilation must avoid Linux libraries
- To do this it needs a custom linking script
- The linking script is /ethos/config/script.ld
- The script only produces static binaries (which don't need run-time linking)

Compilation issues

- Ethos applications are compiled under Linux (on Dom0)
- When the command gcc -o test test.c is executed
 - The pre-processor (cpp) is run on test.c
 - The output of that is fed to the compiler (which includes the compiler front end, optimizer, and assembler)
 - The result compilation is an object file test.o which is then linked against the standard library (libc)
- Procedures and variables in the library are linked against the object file only if needed
- This prevents name conflicts in applications with library functions. (Name resolution occurs left to right with the standard library at the right end)
- And allows the application to substitute their own version in preference to system functions



Compiling a C program for Ethos

• first the gcc command

```
\begin{split} \text{FLAGS} &= -\text{Wall } -\text{fno-builtin } \\ &-\text{fno-leading-underscore} \\ &-\text{fno-stack-protector } -\text{DDEBUG } -\text{g2} \\ \text{INCLUDES} &= -\text{I/ethos/include/} \\ &-\text{I/ethos/include/userspace} \\ \text{gcc } \text{(FLAGS)} \text{ } \text{(INCLUDES)} \text{ } -\text{c } \text{init.c} \text{ } -\text{o} \text{ } \text{init.o} \end{split}
```

- The FLAGS specify all possible warnings, to not do certain types of optimizations, not to allow leading underscores, to provide some form of Address Space Layout Randomization, to define DEBUG to the pre-processor, and to generate debugging information in the .o file
- The include includes the userspace specific include directory and the shared kernel-userspace directory.
- the gcc command runs the first stage of compilation and does not do linking

Jon A. Solworth

Secure OS Design and Implementatio

Userspace

remainder of compilation

Here is what you do after the gcc command

```
Id -T script.ld -nostdlib -o init.elf init.o\
    -L./lib -lestdlib\
    'gcc -print-libgcc-file-name'
        objdump —source init.elf > init.elf.lst
        nm init.elf > init.elf.all.sym
        nm -g init.elf > init.elf.global.sym
```

- the ld line specifies the loader script (script.ld), to not automatically add the standard library and the location of libraries and the use of both the Ethos standard library (estdlib) and the gcc library.
- objdump displays the assembly code from the ELF file
- nm extracts names from the ELF file.

Jon A. Solwort

Secure OS Design and Implementation

Hserspace

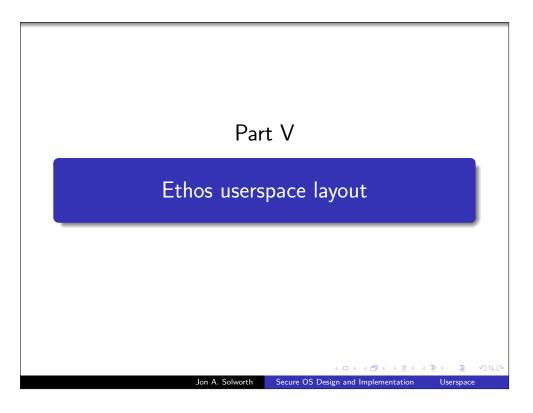
Part IV

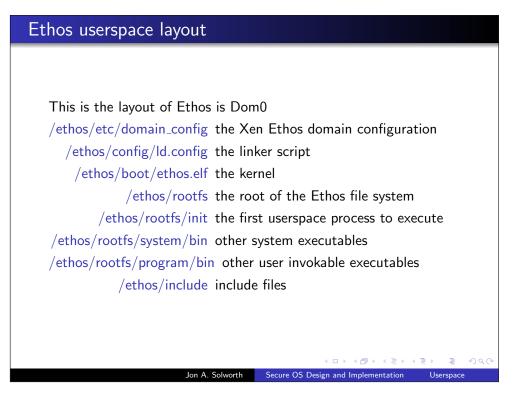
Ethos standard library

Ethos standard library

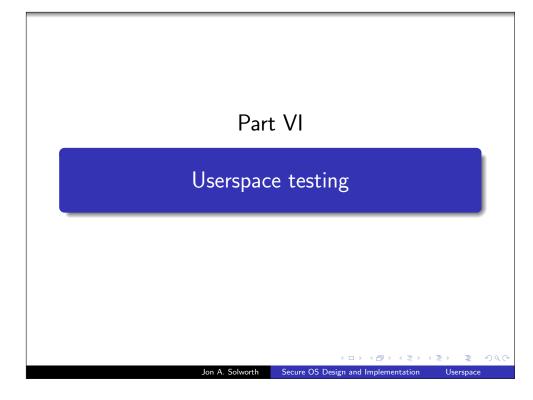
```
readVar Read a file
writeVar Write a file
readStream Read a stream
writeStream Write a stream
getDirectoryFd get the Fd for a directory
getDirectoryVector get a vector of Fd for a directory path vector
tsleep Sleep for specified number of nanoseconds
printf Print to stdout
fprintf Print to fd
```







Creating userspace executables create an init and put it in /ethos/rootfs create some other executables which will be descendants of init These would typically go in /ethos/system/bin We need a standard init Which will start up programs according to a script So that we need to only specify the start up script (and create the necessary programs)



Testing

- ethosUserspace/trunk/test
- Each Makefile has the following targets clean, install, run, check
- first clean, install, and run all tests
- then check all tests
- Do this testing often (after every change)
- Checked in programs should not have regressions
- (Things get more complicated when projects get bigger, > 100K lines)



Jon A. Solworth

Secure OS Design and Implementation

Userspac

Conclusion

- For most OS projects, which produce POSIX interfaces, almost all the work is in building the Kernel
- For non-POSIX interfaces, providing userspace code will be significantly larger than building the Kernel
- To do that we'll have to build libraries
- Port programming languages
- And build applications



Git and Bugzilla use

- Build. check in, and commit to git should be done as often as possible
- Each check in has a comment associated with it
- Checkins also should be synchronized with bugzilla fixes



Jon A. Solwo

Secure OS Design and Implementation

Userspace