THE UNIVERSITY OF UTAH

Verification of Xen tool – (xl console) using SMACK

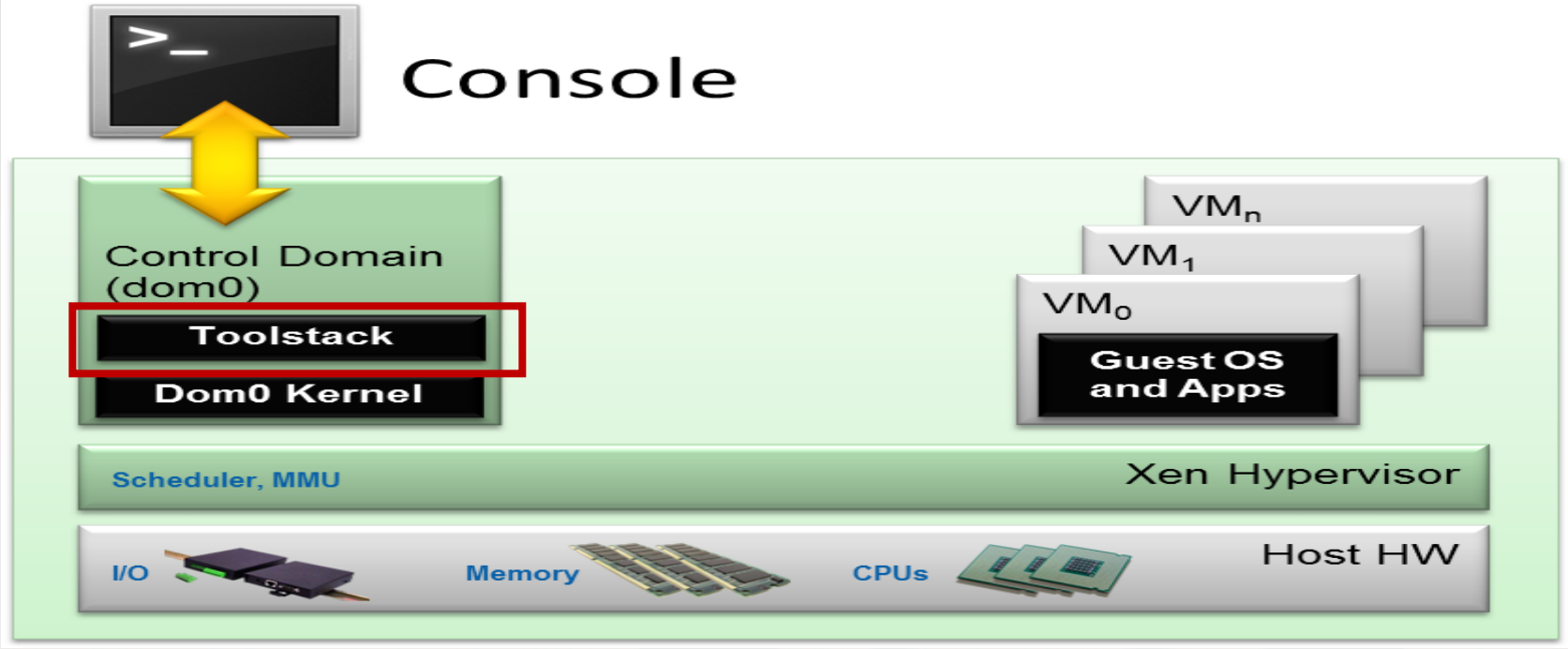
Course: CS6110

Author: Pankaj Kumar

Advisor: Zvonimir Rakamaric

Introduction

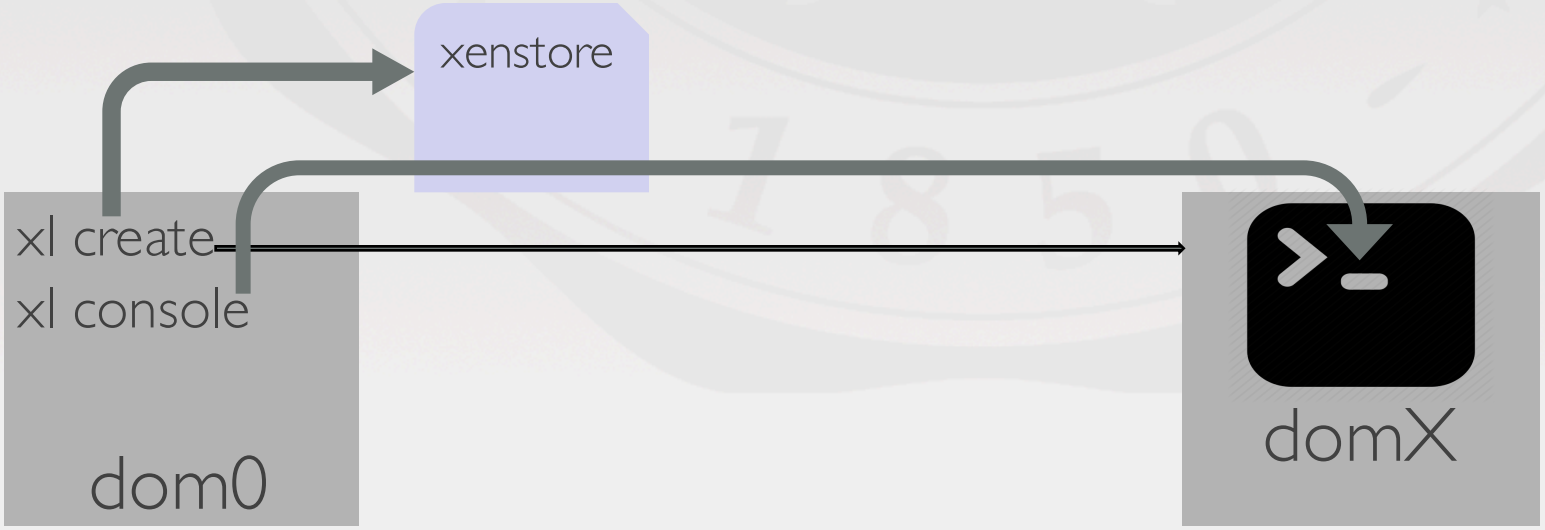
Xen: Type I (bare metal) hypervisor



Domain0 – Privileged virtual machine – hosts Xen tool stack

Xen tools - console

- Connecting console to virtual machines
- Part of Trusted Computing Base
- Written in C
- 2000 lines of code
- Clang compatible



SMACK

- Software Verification tool
- Supports C language
- Handles dynamic memory allocation & pointers arithmetic
- Full support available

Milestones

Feasibility Analysis – Checking Xen compatibility with clang

IR & bit-code generation – Generating IR & extracting bit-code using wllvm

Verification – Running SMACK on bit-code and verifying correctness.

Feasibility analysis

SMACK works on llvm IR

- Used clang 3.7 (SMACK compatibility issues with clang 3.8)

IR & bit-code generation

Generating LLVM bit-code files from C source code files

WLLVM - Compiler wrapper

1. Generates normal object file
2. Generate bit-code file for every object file
3. Stores bit-code files location in a dedicated section in object file itself
4. Concatenates dedicated section's content in one location
5. Generates complete bit-code file from same section using extract-bc tool

Verification

Memory safety

"*smack –memory-safety __bitcode_file__*"

xenconsoled – Coral crash Issue

xenconsole – Issues detected with high loop unrolling factor

- argv issues
- Illegal memory access
- Memory leak issues

Integer overflow

"*smack –signed-integer-overflow __bitcode_file__*"

No bugs detected

Errors & Solutions

Invalid pointer dereference

- Problem : usage(argv[0]);
  Solution: Declare a dummy pointer variable char **argv and allocate memory to it.

- Problem : strtol(argv[x], &end, 10); // x = -1
  Solution: Provided valid index checking.

Errors & Solutions

Memory Leak

- Problem:
```
if (error) { ….
        exit(EINVAL); //Exiting without freeing memory
}
```
Solution: Free the memory allocated so far before exiting
```
if (error) { ….
    free(argv);
    free(end);
    exit(EINVAL); //Exiting after freeing memory
}
```

Unverified paths

- strlen calls
- strtol calls
- FD_ISSET calls

THE UNIVERSITY OF UTAH

SMACK performance

Memory Safety

Binary	Loop unroll factor	Timing (seconds)
xenconsole	4	3300
xenconsole daemon	1	--

Integer Overflow

Binary	Loop unroll factor	Timing (seconds)
xenconsole	4	7.47
xenconsole daemon	1 2	18.4 600 (crash)

Future extension

1.) Verification of xl create tool
2.) xl library verification – One Step deeper

Takeaway

- Xen tool stack is a part of Trusted Computing Base
- Simple code coverage tools are not sufficient e.g. Bullseye
- Using Formal verification tools like SMACK, we can explore all possible paths and expose more bugs

Source Code Location:
https://github.com/pankajk87/XenConsoleVerification.git