Institute for System Programming
of the Russian Academy of Sciences

Instrumenting, Introspection, and Debugging with QEMU

Pavel Dovgalyuk

Institute for System Programming
of the Russian Academy of Sciences

Our projects

- Working on QEMU projects since 2010 (version 0.13)
- Software analysis for x86
- Deterministic replay
- Reverse debugging
- Deterministic replay
- Now working on introspection and debugging projects

Virtual machine introspection

- Extracting data for debugging and analysis
- Semantic gap problem

GDB

- Remote debugging
- Guest system is executed as a single program
- Process information is not available
- Single-stepping may change the execution result

Deterministic and reverse debugging

- Using icount for deterministic timers
- Using checkpoints for faster rewind to the desired moment of execution
- GDB reverse debugging commands
  - reverse-continue, step, next, finish
- Still work-in-progress for mainline QEMU

GDB + scripts

- GDB interacts with QEMU using complex packets
- Conditional breakpoints lead to many VM stops and debugger-QEMU communication
- Very slow for VMI

WinDbg

- Support stealth Windows debugging with WinDbg
- More information than in GDB
- Submitted to qemu-devel

- https://github.com/ispras/qemu/tree/windbg

Native VMI

- Instrumenting guest or TCG code
- Memory access and interrupt callbacks
- Memory and CPU state query interface

QEMU-based VMI frameworks

- PyREBox
- PANDA
- DECAF
- ISP RAS
- and other less mature systems

PyREBox

- PyREBox – Python scriptable Reverse Engineering sandbox
- QEMU 2.10
- Uses Volatility memory forensics
- Python scripting for automated analysis
- Implements interface for mining the VM memory
- https://github.com/Cisco-Talos/pyrebox/

PANDA

- Platform for Architecture-Neutral Dynamic Analysis
- QEMU 2.8.50
- VM introspections
- Taint analysis
- CPU record-replay

- https://github.com/panda-re/panda

DECAF

- Dynamic Executable Code Analysis Framework
- QEMU 1.0
- VM introspection plugins
- Taint analysis

- https://github.com/sycurelab/DECAF

ISP RAS

- Our own approach
- QEMU 2.8.50
- Subsystem for dynamically loaded plugins
- Syscalls and API logging for i386

- https://github.com/ispras/qemu/tree/plugins

VMI requirements for QEMU

- Translation events
- Memory operation events
- Execution events
- Exception events
- Disk and DMA events
- Keyboard and network events
- TLB events
- Monitor commands

Instruction instrumentation

- Instrument at translation
  - Specific instructions
  - Specific addresses
- Get callbacks at execution

Instruction instrumentation

```
0xb7707010: mov %ebx,%edx
0xb7707012: mov 0x8(%esp),%ecx
0xb7707016: mov 0x4(%esp),%ebx
0xb770701a: mov $0x21,%eax
0xb770701f: int $0x80
```

```
---- b770701f 00000000
movi_i64 tmp13,$0xb7707020
movi_i64 tmp14,$0x7fef9a788670
call start_system_call, $0x0,$0,tmp13,tmp14
movi_i32 tmp3,$0xffffffffb770701f
st_i32 tmp3,env,$0x20
movi_i32 tmp11,$0x2
movi_i32 tmp12,$0x80
call raise_interrupt, $0x0,$0,env,tmp12,tmp11
set_label $L0
exit_tb $0x7fef8e6dca13
```

TCG Instrumentation

- Platform-independent instrumentation
- Used for taint analysis in DECAF and PANDA
- Not complete because of helpers

Memory instrumentation

- Memory ops performed through softmmu-callbacks and translated code
  - DECAF supports only callbacks
- Memory forensics through exported load functions

Memory instrumentation

- Logging
- Cache simulator
- Forensics
- Anomalies detection

Interrupts and exceptions

- Only asynchronous callbacks
- Logging peripheral interrupts
- Detecting page mapping

Instrumentation/introspection applications

- Logging syscalls
- Logging API
- Logging memory accesses
  - for cache simulator
  - for debugging the firmwares

QEMU instrumentation

- 10+ attempts to add instrumentation API
- Does it have to be included into mainline?
- QEMU-VMI interface is very narrow
  - ~20 callbacks
  - ~50 externally accessible functions