

Repeatable Reverse Engineering with PANDA

B. Dolan-Gavitt (NYU), J. Hodosh (MIT), P. Hulin (MIT), T. Leek (MIT), R. Whelan (MIT)

**Slides adapted from RECON 2014 presentation, B. Dolan-Gavitt*

What is PANDA?

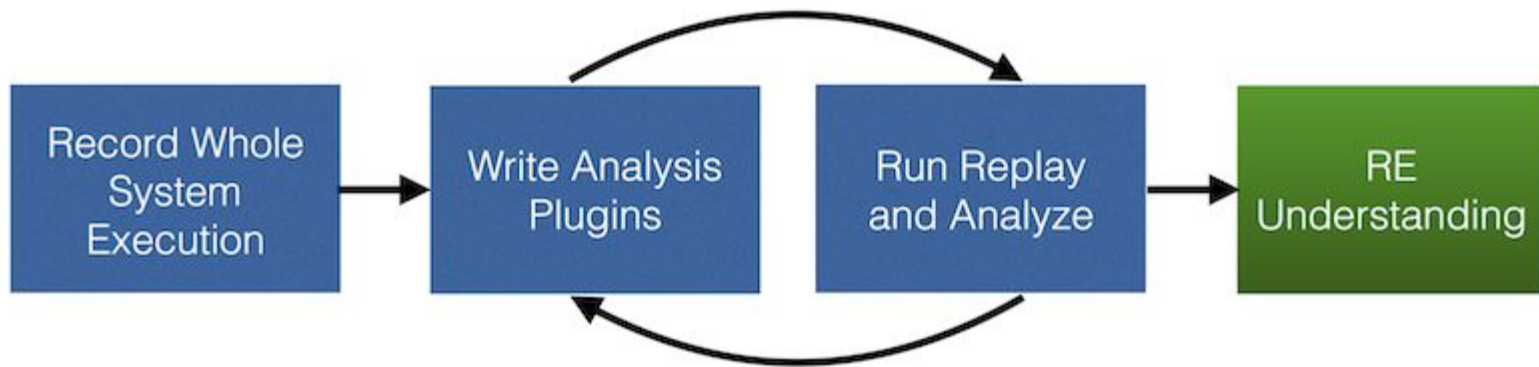
Platform for **A**rchitecture **N**eutral **D**ynamic **A**nalysis

- Open source dynamic analysis tool to support whole system reverse engineering
- Functionality can be extended by analysis plugins
- Supports x86, ARM, MIPS

What Makes PANDA Different?

- Ability to 'Record and Replay'
- Capability to repeat execution trace with all the same data over and over again
- Analyze execution (offline) in order to develop understanding of the system

Basic PANDA Workflow

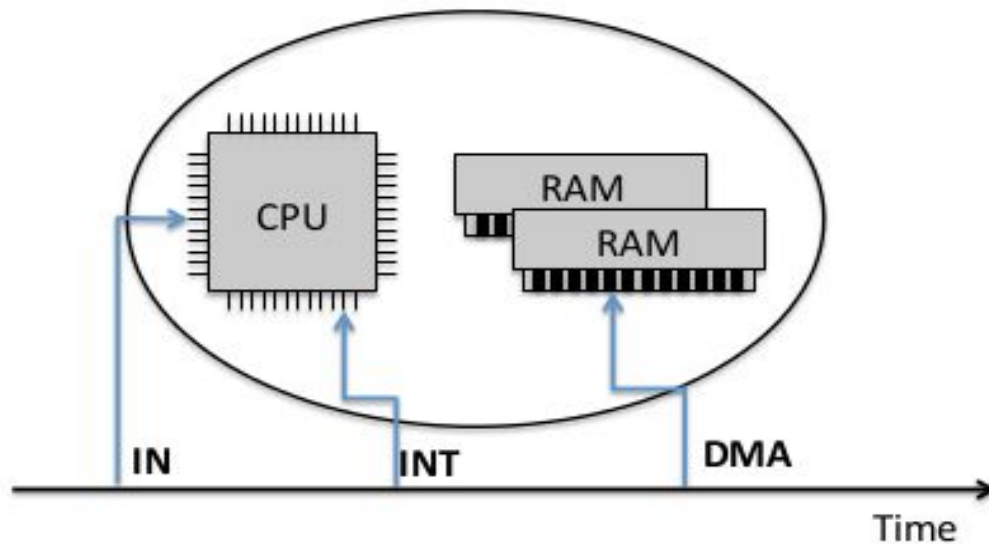


<https://github.com/moyix/panda/blob/master/docs/manual.md>

Record and Replay

- Draw “imaginary line” around CPU and memory in Guest OS
- Log three kinds of input that cross the line:
 - Data entering the CPU
 - Hardware interrupt and parameters
 - Data written to RAM during direct memory access
- Also record ‘trace point’ on any write to the log (determines when to replay input)
 - Value of program counter, instruction count, and implicit loop variable

Record and Replay



*“Repeatable Reverse Engineering with PANDA”,
B. Dolan-Gavitt (NYU), J. Hodosh (MIT), P. Hulin (MIT), T. Leek (MIT), R. Whelan (MIT)*

Record and Replay

- Different from replay in VMWare
- PANDA does not execute any device code: “no need to ‘go live’”
- Intent is to perform specific replay-based analysis
- Allows for in-depth analysis too slow to run in real-time
 - e.g., taint flow analysis

Record and Replay

- Allows for repeatability of results (difficult to do with dynamic analysis)
- Traditional dynamic analysis performed in a debugger cannot execute backwards
- Inspection of earlier program state requires program restart:
 - Heap address will change
 - Data structure location will need to be found again (w/o debugging symbols)
- PANDA record and replay accelerates the reverse engineering process

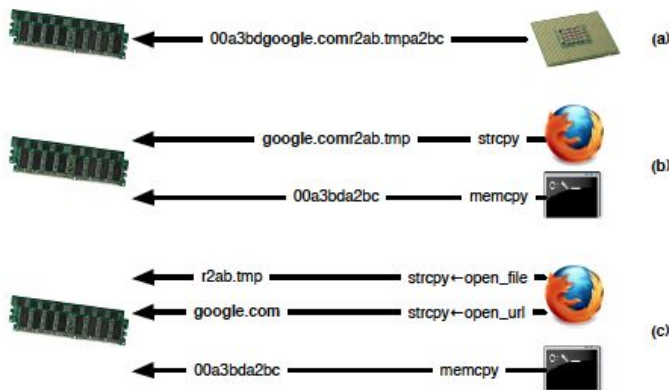
Sharing of Replays

- Size of recordings are small enough to be shared between analysts
- Replays are posted at www.rrshare.org and can be downloaded (.rr files)

Plugin Architecture

- Plugins allow for more specific analysis after whole-system recording
- Implement functions that can take action at various instrumentation points
- Examples include:

-Tappan Zee (North) Bridge (TZB): set of techniques to mine memory access



*“Repeatable Reverse Engineering with PANDA”,
B. Dolan-Gavitt (NYU), J. Hodosh (MIT), P. Hulin (MIT),
T. Leek (MIT), R. Whelan (MIT)*

Plugin Architecture

-Other examples include:

- callstack_instr: calling context tracking

- syscalls: system call tracking

- taint2: data labeling and taint flow tracking

- scissors: excise smaller portions of replay to focus attention

Plugin Architecture

- QEMU code execution process:

 - Guest OS code is translated to internal representation (TCG IR)

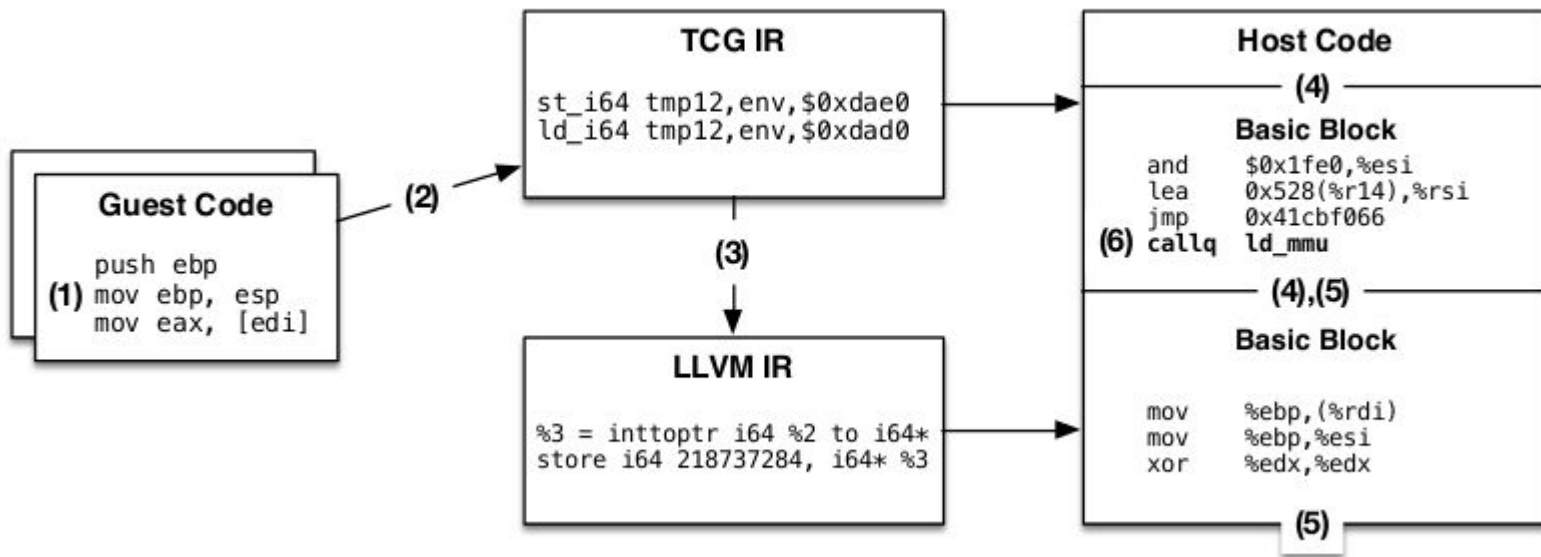
 - QEMU generates basic block of code from IR directly executable on host

- PANDA takes TCG IR and generates LLVM instructions to execute on host

- LLVM translation allows for platform independence, more complex taint analysis

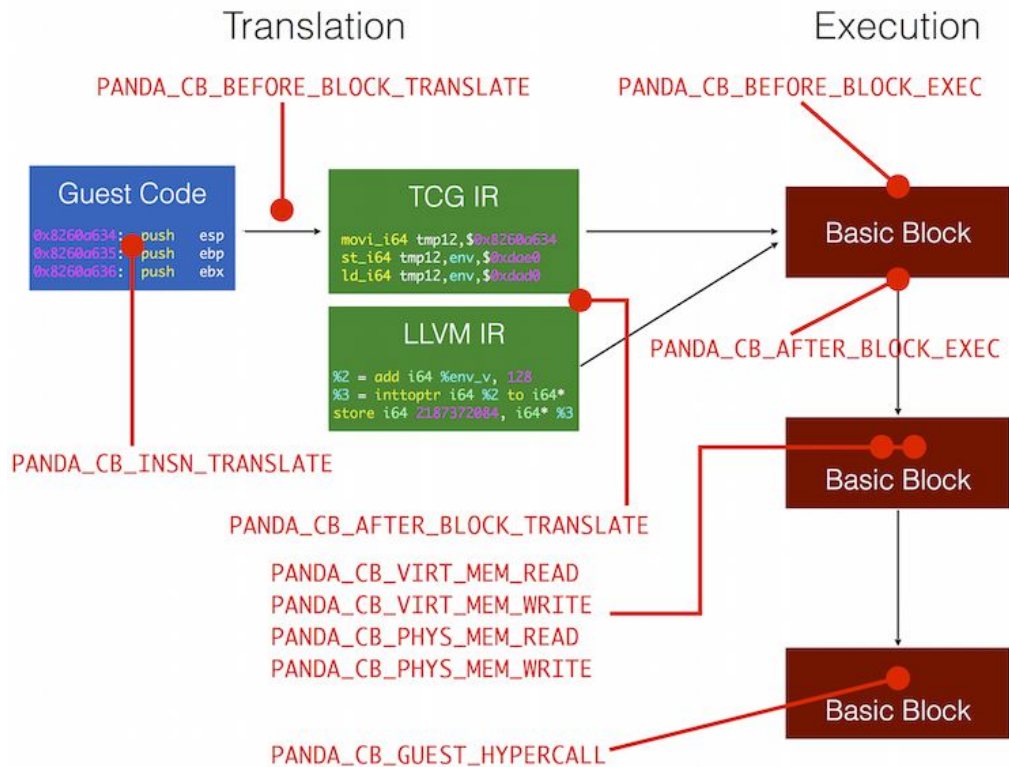
- Plugins allow user to register callbacks at various points of QEMU execution

Plugin Architecture



*"Repeatable Reverse Engineering with PANDA",
B. Dolan-Gavitt (NYU), J. Hodosh (MIT), P. Hulin (MIT), T. Leek (MIT), R. Whelan (MIT)*

Plugin Architecture



Plugin Architecture

- PANDA framework allows for complex plugin-to-plugin interaction:
- Example usage: (e.g., detect if user's password is sent out over the network)
 - search tap points for string which defines a callback that triggers on match
 - analysis plugin might use this callback to be notified when this string is used
 - taint plugin could then taint the string in memory and register a callback with the syscalls plugin that triggers whenever `sys_send` is called (and determines whether the data written is tainted)

Case Studies

- Successfully used PANDA to quickly reverse engineer a popular video game from 1999 that required a 26-character CD-key
- Demonstrated breaking of Spotify DRM
- Rapidly identified cause of crash in Internet Explorer as a 'use-after-free' bug (specifically CVE-2011-1255)

Conclusion

- Primary limitation is performance (record-time)
- Advantage over traditional dynamic analysis is repeatability of results using record/replay capability
- Compelling use cases have been presented for employing PANDA to rapidly reverse engineer complex binary systems
- Application to reverse engineering malware

References

B. Dolan-Gavitt, J. Hodosh, P. Hulin, T. Leek, R. Whelan, *Repeatable Reverse Engineering with PANDA*, Proceedings of the 5th Program Protection and Reverse Engineering Workshop, Article No. 4, December 2015

B. Dolan-Gavitt, T. Leek, *Tappan Zee (North) Bridge: Mining Memory Accesses for Introspection*, November 2013

R. Whelan, *Architecture-Independent Dynamic Information Flow Tracking*, Master's Thesis, Northeastern University, November 2012.

PANDA manual, <https://github.com/moyix/panda/blob/master/docs/manual.md>