Structure-aware Fuzzing of Oxenstored

The Dawn of a Security Audit

Edwin Török, Citrix R&D Limited

What is the xenstore daemon

- a key-value tree with ACLs and transactions containing virtual device and guest metadata
- part of guest virtual device ABI
- any modifications to xenstore daemon need to retain compatibility with existing guests (we can't crash old guests on live-migrate or boot)
- 2 flavours: C and OCaml, both used in production in large deployments

Where is the spec?

- Documentation of existing behaviour
- We have a document describing the existing behaviour of xenstored
- Not a formal spec, implementation differences might exist between C and oxenstored, and transaction semantics not fully specified
- Aim is to have agreement between C and OCaml implementations

The tale of XSA-115 and restartability

- Got asked to help fix one bug in oxenstored and implement a way to live-update oxenstored
- Rabbithole: discovered/had to fix 6 other bugs while fixing the one XSA and implementing restartability
- Why? I tried to come up with a way to test live-updates. Particularly difficult in a security update, has to be automated
- State of testing of xenstored not great ...

Testing?

- Very little unit tests, not run automatically
- Testsuite in XenServer and elsewhere exercises it from a system-level: failures hard to triage and long feedback cycle
- Pre-restartability had to reboot host everytime wanted to test a change to oxenstored: development too slow
- Very little development done on oxenstored, still haven't switched to mirage version of xenstored
- Structure-aware fuzzing can be used to write a minimal unit test and allow some minimal testing done on devbox, not requiring Xen

Structure-aware fuzzing

- Classic fuzzing: afl. Generate random data in parallel -> does it break the application?
  - May not exercise meaningful input/semantics
- Classic property testing: QuickCheck. Boolean property describing desired behaviour semantics -> does it hold on random data?
  - Typically single-threaded, slow to find bugs beyond a limit
  - Great for testing on meaningful input, and providing developer feedback

- Structure aware fuzzing: drive the random number generator in QuickCheck from afl
  - Advantages of both, no downsides
  - Can run in "pure quickcheck mode" for unit testing with fixed seed and limit
  - Can be parallelized with afl and typically finds at least one bug in 30m on a 24-core box
- Various implementations in OCaml: qcstm, crowbar, monolith

State machines

- Simple concept: run 2 oxenstored in parallel, feed same commands to both and do live-update on just one of them: behaviour after live-update must be equivalent
- 2 state machines: reference and implementation under test
- reference = implementation that ignores live-update commands
- implementation under test = accepts live update commands
- Both run in same process, without requiring Xen in "mock" mode
- Input: command packets built up from primitives (random string, random int, random choice)

State machines

- Other possibilites in the future: different versions, C vs O
- Check that desired properties hold: security and equivalent semantics
- When bug is found shrink testcase to minimal command packets and print base64 encoded input
- Replay mode: given a small base64 input replay the commands, useful for debugging and fixing bug: just replay seed that triggered bug until fixed on updated codebase
- afl-cmin can be used to shrink testcases

Next steps

- This is just the dawn of a security audit of xenstore
- fuzzer only capable of finding semantic bugs so far, though some semantic bugs are security bugs too:
  - quota is different after live-update: replaying a tree and building it up from primitive ops is not equivalent and can be used to ignore quotas
  - while fixing semantic bugs found by fuzzer discover security bugs because I happen to read surrounding code and wonder where all the security protections are (e.g. you could delete / in oxenstored without any permission checks, fixed now!)

Next steps

- more properties could be written, e.g. to check transaction semantics, basic sanity test of operations
- reference implementation could be a simpler/ideal version of oxenstored based on "the spec" (started on this, not yet in usable state).
- Ideal: executable specification as property test

Benefits to Xen community

- There is development activity on oxenstored again!
- Modernized build system based on Dune (optional, only if you want to use the fuzzer)
- Shorter feedback cycle to developer: silly bugs that cause oxenstored to fail on startup can be caught early by unit test
- Most of the time do not even need to copy and test on a box: semantic bugs can be fixed with the help of the unit test
- *If* I need to test on a box I can use live-update to avoid rebooting the box (caveat: still needs some patches to make it work reliably)
- Patches available on xen-devel, testing and review underway

Some example bugs

Demo

Q&A