OFF: Bugspray for Openflow

Ram Durairajan, Joel Sommers, Paul Barford
Motivation

- Debugging SDN applications is hard
- “Runs as designed” may be insufficient
- Deployments must cope with wide range of operating conditions
- How can we answer the following question:

Will my SDN app run as designed when deployed in a live setting?

Our Solution: Oфф!

rkrish@cs.wisc.edu
Design Goals of OFf

• A debugging and test environment for SDN developers
• Default debugging options
  • Stepping, breakpoints, watch variables, etc.
• Comprehensive testing for SDN applications
  • Packet replay, packet tracing, visualization, alerts, etc.
• Tie unwanted network behavior to controller
• Simple, light-weight and no hardware support
• Facilitate transition to live environments

rkrish@cs.wisc.edu
Related work

• Debuggers
  – ndb (Handigol et al., 2012)
  – NetSight (Handigol et al., 2014)

• Replay tool
  – OFRewind (Wundsam et al., 2011)

• Static analysis and symbolic execution tools
  – Veriflow (Kurshid et al., 2013)
  – Header Space Analysis (Kazemian et al., 2012)
  – NICE (Canini et al., 2012)
OFFf Architecture

User Interface Wrapper

- Debugger
- Trace Replay
- Diff Report Generator

Keyboard

Screen

POX Controller

fs-sdn Simulator

rkrish@cs.wisc.edu
OFFf Commands

- longlist and shortlist source code
- pretty print expressions
- hide and unhide frames
- interactive interpreter with all variables in scope
- track, watch, or unwatch variables
- edit source files during debugging
- enable or disable break points on the fly
- sticky mode to visualize code
OFF Additional Features

• Trace packet through the network
  – *Holistic* view of flows, controller and switches
  – No additional hardware

• Replay packets later
  – No OFP modification

• Defect configuration changes
  – Topology changes
  – Rule/action changes
  – Performance variations
OFFf in Action

- We demonstrate OFFf in three scenarios
  - Incorrect ordering of updates
  - Bad multi-app interaction
  - Unexpected rule expiration
- Goal: Identify logical bugs in the source code that lead to transient outages and losses
Incorrect Ordering of Updates

Simple Routing App

POX Controller

Switch A

1s

Switch B

7s

Switch C

1s

Host 1

Host 2

rkrish@cs.wisc.edu
Solution: Incorrect ordering

• Installation order - C, B, and then A
• Handle barrier messages

• Using OFF
  – Replay packets
    • find packets that are dropped at B as rules are not installed
  – Set a break point => sticky mode => watch at B
  – Infer ordering problem and fix
  – Trace and Diff Reports to verify fix
Bad Multi-app Interaction

Block: 10.0.0.1 to 10.0.0.4

Modify: From: 10.0.0.1
SrcIP: 10.0.0.2
Modify: To: 10.0.0.3
DstIP: 10.0.0.4
Allow: 10.0.0.2 to 10.0.0.3

rkrish@cs.wisc.edu
Solution: Bad Multi-app Interaction

- Using Off developer 2 can
  - collect network traces (T1)
  - prototype routing app using fs-sdn
  - collect traces again (T2)
  - runs diff reports (T1 and T2)
- Rule set conflicts are found
- Change and iterate
- Verify firewall invariants
Conclusion

• Oфф – a debugging and test environment for SDN developers
• Oфф is simple, flexible, and light-weight
• We demonstrate Oфф using three scenarios
Thank you!

Source Code
https://github.com/52-41-4d/fs-master

Questions?

rkrish@cs.wisc.edu
Unexpected Rule Expiration

Simple Routing App

POX Controller

Switch A
Host 1
Switch B
Switch C
Host (Untrusted)
Switch D
10.0.0.0/8
10.5.0.0/16
t=30
Host (Trusted)
Solution: Unexpected Rule Expiration

• Using OFF
  – prototype using fs-sdn and replay trace
  – trace flow and rules
    • wrong rule triggered
  – Change the timeout behavior
  – Verify using diff reports