Detecting TCP regressions with tcpdiff

NYCBSDCon 2008

Mike Silbersack
silby@silby.com

http://www.silby.com/nycbsdcon08/

Presentation Overview

- What is tcpdiff
- The tcpdiff test apparatus
- How tcpdiff analysis works
- Four TCP bugs tested
- Systems that can not be tested
- Future Work

What is tcpdiff?

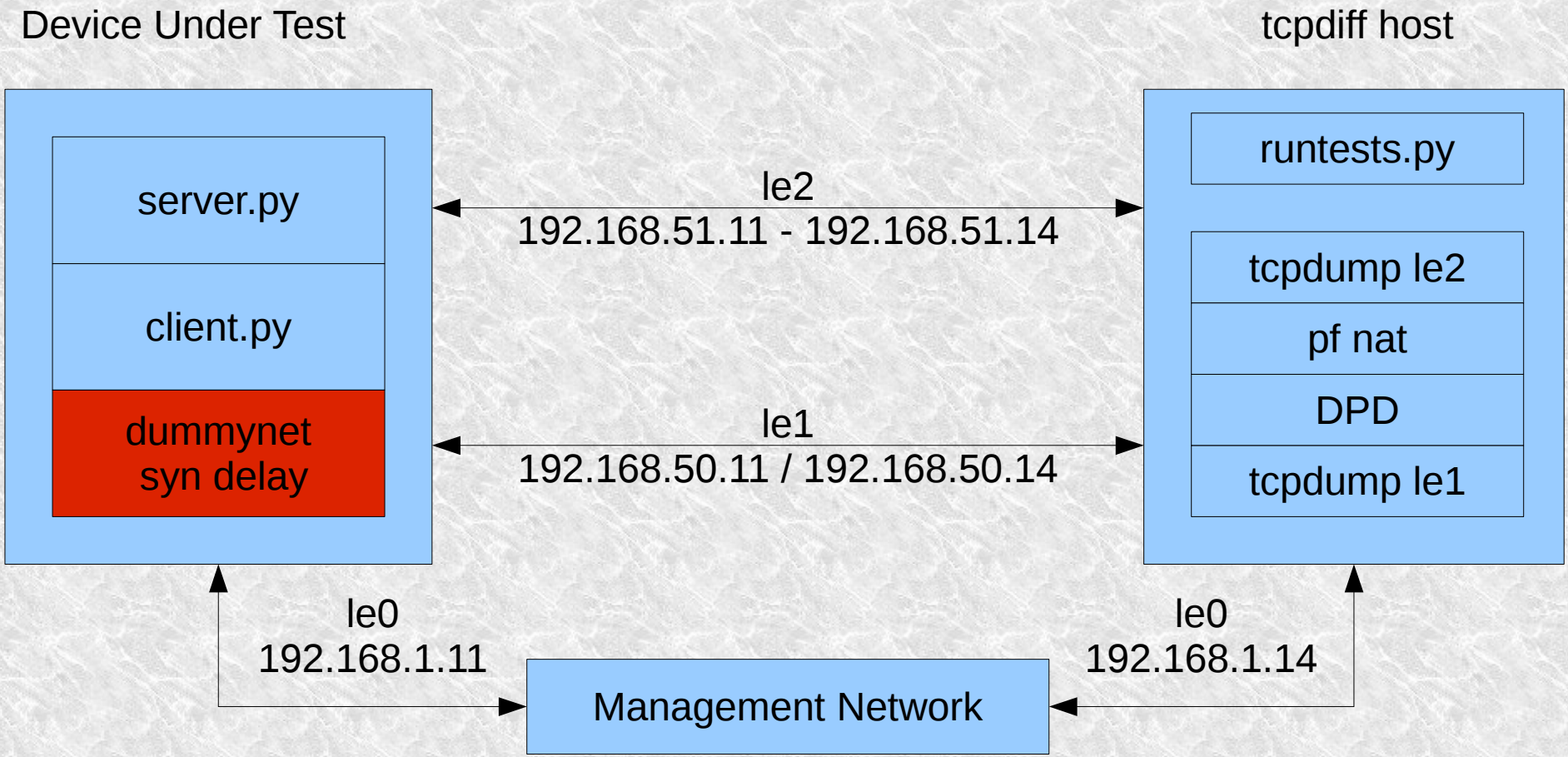
tcpdiff is designed to automatically detect differences in TCP behavior between different versions of an operating system and display those differences in an easy to understand format.

tcpdiff looks only at on the wire behavior of the network stack. The kernel under test requires no modifications; all faults are injected on the wire.

What is tcpdiff not?

tcpdiff is not designed to know whether a TCP stack operates correctly or not. All it can do is compare two TCP stacks. The value judgment of whether a certain change between version X and Y of a TCP stack is good or bad is left to human eyes.

The tcpdiff test apparatus



Tools used in the apparatus

- VMware
  - Physical hardware would be better, of course.
- DPD (Deterministic Packet Discard)
  - Lawrence Stewart's modification to dummynet that allows specific packets to be dropped
  - http://caia.swin.edu.au/urp/newtcp/tools.html
- OpenBSD's pf
  - Performs the NAT magic to send packets back to the Device Under Test.

Tools used in the apparatus, continued

- Dummynet
  - Used to delay the initial SYN packet so that the SYN and SYN-ACK retransmission timers are not synchronized.
  - This should be resident on the tcpdiff host, but stacking multiple dummynet pipes caused buggy results.
- Tcpdump
  - This is run on both interfaces so that the correct behavior of DPD can be observed.

Tools used in the apparatus, continued

- server.py
  - Accepts a connection, sends 64K of data, closes the connection.
- client.py
  - Initiates a connection, does a MSG_WAITALL recv to read all data at once, sleeps one second, closes the connection. Waits for the FIN_WAIT/etc socket to clear before exiting.
    - The MSG_WAITALL read is important as it prevents window updates from showing up on the wire.

Tools used in the apparatus, continued

- runtests.py
  - Set up various sysctls
  - Start server.py
  - Loop for each drop pattern
    - Reset the hostcache
    - Reset the DPD rules
    - Reset tcpdumps
    - Run client.py
- test_allkernels.py
  - Runs runtests.py in a loop over all test kernels in /boot.

Drop patterns

- 64K of application data sent by server
  - 27 packets, 1416 bytes of packet data from client to server
  - 49 packets, 68096 bytes of packet data from server to client
- Drop patterns tested:
  - A mixture of single and double packet drops were tested, focusing on the start and end of each connection.

Drop patterns enumerated

Drops are of the form (x, y), where x is the client packet dropped, and y is the server packet dropped.

Single drops:
(0, 1) (0, 2) (0, 3) (0, 4)
(0, 44) (0, 45) (0, 46) (0, 47) (0, 48) (0, 49)
(1, 0) (2, 0) (3, 0) (4, 0)
(24, 0) (25, 0) (26, 0) (27, 0) (28, 0) (29, 0)
Double drops:
(1, 1) (1, 2) (1, 3) (1, 4)
(2, 1) (2, 2) (2, 3) (2, 4)
(3, 1) (3, 2) (3, 3) (3, 4)
(4, 1) (4, 2) (4, 3) (4, 4)
(24, 44) (24, 45) (24, 46) (24, 47) (24, 48) (24, 49)
(25, 44) (25, 45) (25, 46) (25, 47) (25, 48) (25, 49)
(26, 44) (26, 45) (26, 46) (26, 47) (26, 48) (26, 49)
(27, 44) (27, 45) (27, 46) (27, 47) (27, 48) (27, 49)
(28, 44) (28, 45) (28, 46) (28, 47) (28, 48) (28, 49)
(29, 44) (29, 45) (29, 46) (29, 47) (29, 48) (29, 49)

How tcpdiff analysis works

- Stage 1: process.py / normalize.py
  - Normalize output
    - Normalize IP/port into “client > server” and “server > client”
    - Zero out TCP timestamps, tcpdump timestamps
    - Normalize ISNs
  - Split output into directional flows
  - Add notation as to which packets were dropped by DPD.

How tcpdiff analysis works, continued

- Stage 2: compare.py
  - Runs diff on two processed directories.

Test Results: 12/5/07 to 12/3/07

- Show 1205to1203.txt

Test Explanation: 12/5/07 to 12/3/07

- SACK negotiation
  - Break
    - tcp_syncache.c 1.105 Mar 15, 2007
  - Fix
    - tcp_syncache.c 1.136 Dec 4, 2007

Test Results: 9/8/08 to 9/6/07

- Show 0908to0906.txt

Test Explanation: 9/8/08 to 9/6/07

- Tcp timer merge/unmerge
  - Break – Apr 11, 2007
    - tcp_timer.c 1.90
  - Fix – Sep 7, 2007
    - tcp_timer.c 1.96
- What improvement needs to be made to tcpdiff to detect this?
  - A program that holds a socket open after doing a shutdown may emit extra packets.

Test Results: 8/1/07 to 7/30/07

- Show 0801to0730.txt

Test Explanation: 8/1/07 to 7/30/07

- FreeBSD 6-current – kern.hz 100 -> 1000
  - Break - ?
  - Fix – tcp_timer.h 1.37 Jul 31, 2007
- Two improvements to tcpdiff could detect this
  - Add time based drops to DPD
  - Detect packet timing in tcpdiff.

Test Results: 7/29/07 to 7/27/07

- Show 0729to0727.txt or 0729to0727b.txt

Test Explanation: 7/29/07 to 7/27/07

- FreeBSD 7-current – SYNcache retransmission
  - Break
    - tcp_syncache.c 1.87 Jun 17, 2006
  - Fix
    - tcp_syncache.c 1.126/1.127 Jul 28, 2007

Systems that can not be tested

- FreeBSD < 4.0.
  - Delayed acks in these versions of FreeBSD are run from a global timer, and as such are unpredictable.
  - FreeBSD 4+ use per-connection timers; delayed acks are predictable
- FreeBSD < 7.0.
  - The socket copyin routine hands off data from userspace to socket buffers in mbuf cluster (2k) sized chunks, leading to a condition where less than mtu sized packets are sent if the tcp output routine gets ahead.

A run with FreeBSD 6.3

IP server > client: . ack 1 win 64074 <nop,nop,timestamp 0 0>
IP server > client: . 1:1449(1448) ack 1 win 64074 <nop,nop,timestamp 0 0>
IP server > client: . 1449:2897(1448) ack 1 win 64074 <nop,nop,timestamp 0 0>
IP server > client: P 2897:4097(1200) ack 1 win 64074 <nop,nop,timestamp 0 0>
IP server > client: . 4097:5545(1448) ack 1 win 64074 <nop,nop,timestamp 0 0>
IP server > client: P 5545:6145(600) ack 1 win 64074 <nop,nop,timestamp 0 0>
IP server > client: . 6145:7593(1448) ack 1 win 64074 <nop,nop,timestamp 0 0>
IP server > client: P 7593:8193(600) ack 1 win 64074 <nop,nop,timestamp 0 0>

The pattern of full vs less than full sized frames differs nearly each time the test is run.

Future Work

- Run tcpdiff nightly!
- Move the SYN delay off of the device under test and on to the tcpdiff host.
- Fix the “dropped” packet notation; it is incorrect when an identical packet is retransmitted, such as the case when a SYN packet is dropped.
  - This does not affect comparison results, although it makes the results harder to read.

Future Work continued

- Add more test cases
  - Client connects, sends data to server.
  - Client connects to echo server, sends strings at .1 second intervals. Disconnects once final string has been echoed.
  - Http fetch of a file
- Add more drop patterns
- Determine which other OSes can be tested

Future Work continued

- Make the report friendlier
- Generate kernels representing more fixed bugs, verify that tcpdiff can detect them.
- Try creating a three VM setup so that the device under test is actually ipfw or pf rather than the host TCP stack.
- Backport socket buffer optimization to FreeBSD 6 so that it can be tested

Questions?

Notes on VMware

- Check your clock accuracy!
  - Disable SpeedStep / Cool & Quiet
    - Install AMD “Dual Core Optimizer” on Athlon X2 systems, or the TSCs of the two cores desync. Applies even if the guest is not using the TSC.
  - Set tickrate down to 100hz
    - Note that this changes timer accuracy to 10ms
- Verify your clock
  - ntpdate pool.ntp.org ; sleep 10 ; ntpdate pool.ntp.org
  - You should see only sub-second differences