TCP/IP Regression Test Suite

Shivansh Rai

Introduction

Overview

Regression testing is one of the most critical elements of the test artifacts and proves to be one of the most preventive measures for testing a software. Currently, within FreeBSD, there is no such tool to perform regression testing of the TCP/IP network stack. The purpose of this project is to develop tests using a regression testing tool which can then be integrated with FreeBSD. Once integrated, the tool will also facilitate further development of such tests. The regression testing tool of choice here is `packetdrill`.

About `packetdrill`[1]

`packetdrill` is an open-source scripting tool that enables testing the correctness and performance of entire TCP/UDP/IP network stack implementations, from the system call layer to the hardware network interface, for both IPv4 and IPv6.

About `netperf`[2]

`netperf` is a software application that provides network bandwidth testing between two hosts on a network. It supports Unix domain sockets, TCP, SCTP, DLPI and UDP via BSD sockets.

Potential Mentor

Hiren Panchasara (hiren@freebsd.org)

---

[1] https://github.com/google/packetdrill

[2] http://www.netperf.org/netperf/

The problem

Despite their importance in modern computer systems, network protocols often undergo only ad-hoc testing before their deployment, and thus they are often tedious to troubleshoot on failure. A regression testing tool is required to help in various scenarios/phases such as :

- Black box testing for new feature development.
- Need for more precision in load tests than `netperf`.
- Need for replaying traces to reproduce issues while troubleshooting.
- Testing other aspects such as correctness, performance and security of network stacks.

Currently, there is no such regression testing tool for FreeBSD which facilitates development of new regression tests.

Project Abstract

`packetdrill` currently supports testing multiple scenarios for TCP/IP protocol suite within Linux. This project aims to design and implement a wire level regression test suite for FreeBSD using `packetdrill`. The test suite will exercise various states in the TCP/IP protocol suite, with both IPv4 and IPv6 support. Besides Linux, the `packetdrill` tool works on {Free, Net, Open} BSD.

The existing Linux test suite implemented within `packetdrill` will provide a basis for understanding, and implementation of the FreeBSD test suite. For the current scope of the project, only a subset of the existing test scenarios will be implemented.

Why Packetdrill?

While valuable for measuring overall performance, TCP regression testing with `netperf`, application load tests, or production workloads can fail to reveal significant functional bugs in congestion control, loss recovery, flow control, security, DoS hardening and protocol state machines. Such approaches suffer from noise due to variations in site/network conditions or content, and a lack of precision and isolation, thus bugs in these areas can go unnoticed. Since `netperf` is supposed to be more for benchmarking purposes and what we are trying to do is measure correctness, `packetdrill`, which was built with the same mindset, seemed an apt choice for this project.

Technical Details

The tests made using `packetdrill` will be initially based on scenarios taken from the linux tests. One such scenario is testing the TCP stack when the injected packet has both SYN and RST bits set. In this case, the TCP stack should not respond to the incoming RSTs, or else we could get infinite RST ping-pong storms. The success of this test will depend on whether we can still successfully establish a connection after a valid SYN is injected into the stack.

With the initial contributions done previously, I aim at testing the FreeBSD TCP stack behaviour by setting combinations of various TCP flags while injecting packets into the system under test (SUT).

Test Plan

`packetdrill` supports two modes of testing - local and remote. A TUN virtual network device is used in the local testing and a physical NIC is used for remote testing.
Local testing is relatively easier to use because there is less timing variation and the users need not coordinate access to multiple machines.

To avoid conflicts arising due to memory locking used in `packetdrill`, the following command must be run on a FreeBSD machine -

```
>> sudo sysctl -w vm.old_mlock = 1
```

Or following line should be placed in `/etc/sysctl.conf` -

```
vm.old_mlock = 1
```

Local mode testing

Local mode is the default mode, and hence the user need not specify any special command line flags.

```
>> ./packetdrill -v <test-script.pkt>
```

Executing the above command will give the information about the inbound injected and outbound sniffed packets which can be studied and checked whether in accordance with the expected behaviour. The TUN virtual network device will be used as a source and sink for packets in this case.

Remote mode testing

On the system under test (i.e the “client” machine), a command line option to enable remote mode (acting as a client) and a second option to specify the IP address of the remote server machine to which the client packetdrill instance will connect must be specified.

```
client>> ./packetdrill --wire_client --wire_server_ip=<server_ip>
<test-script.pkt>
```

On the remote machine, using the same layer 2 broadcast domain (same hub/switch), a packetdrill process acting as a “wire server” daemon to inject and sniff packets remotely on the wire will be started.

```
server>> ./packetdrill --wire_server
```

The client instance will connect to the server (using TCP), and will send command line options and contents of the script file. Then, the two packetdrill instances will work in coherence to execute the script and test the client machine’s network stack.

IPv4 and IPv6 protocol testing

`packetdrill` supports IPv4, IPv6 and dual-stack modes. The modes can be specified by the user with `--ip_version` command line flag. To get FreeBSD to allow using ipv4-mapped-ipv6 mode, the kernel must be notified with the following command -

```
>> sysctl -w net.inet6.ip6.v6only = 0
```

For testing using AF_INET6 sockets with IPv4 traffic -

```
>> ./packetdrill --ip_version=ipv4-mapped-ipv6 <test-script.pkt>
```

For testing using AF_INET6 sockets with IPv6 traffic -

```
>> ./packetdrill --ip_version=ipv6 --mtu=1520 <test-script.pkt>
```

Since the IPv6 headers are 20 bytes larger than the IPv4 headers, the MTU has to be set to 1520 to address the extra 20 bytes, rather than the standard size of 1500 bytes.

Deliverables

- Development of TCP/IP based test suite for FreeBSD using `packetdrill`.
- Attempt at covering all the scenarios implemented in `packetdrill` for Linux.
- If all the existing scenarios from Linux have been covered, attempt at working on new scenarios.
- Attempt to create tests based on UDP and those related to sockets.
- `packetdrill` currently supports testing only a single connection at a time. An attempt will be made to patch it to support multiple concurrent connections.
- The current remote mode available in `packetdrill` allows testing a remote host provided there is already an instance of `packetdrill` running on it. There is not yet support for testing a remote host that does not have `packetdrill` running. One such approach for enabling support for this can be that instead of getting command line arguments and the script over a TCP connection, the current instance can get it directly. Hence, the logic for handshake with the client will be removed, the packets will be injected and the client will wait for inbound packets.

List of scenarios to be covered

The following scenarios will be covered initially -

1. Three-way handshake **(done)**
2. Reset from closed state
3. Reset from non-synchronized state
4. Reset from synchronized state
5. TCP options establishment **(5 tests contributed)**
6. Sliding window protocol
7. Urgent pointer
8. Selective acknowledgements
9. TCP timestamps
10. Time-wait configuration
11. Connection close
12. Simultaneous close
13. Receive ACKs, RSTs, and URGs while window is zero
14. Receive window size advertisement
15. Transmit window size advertisement
16. Support partner shrinking window
17. Silly window syndrome avoidance
18. Zero window handling
19. Zero window probing

Project Schedule

Start	End	Task
23 May		Start of coding
23 May	24 May	Checking for compatibility of previously developed tests for Linux with FreeBSD
24 May	19 June	Manual development of tests based on TCP, considering all the scenarios covered in Linux tests
20 June	27 June	Mid-term Evaluations
28 June	14 July	Attempt at developing new tests based on UDP and socket based tests for FreeBSD
15 July	31 July	Attempt at patching `packetdrill` by adding a new mode of testing in which the remote host will not need an instance of `packetdrill` running
1 Aug	11 Aug	Attempt at patching `packetdrill` to support multiple concurrent connections
12 Aug	14 Aug	Code review
15 Aug		End of coding (soft)
23 Aug		End of coding (hard)

The Code

https://github.com/shivrai/packetdrill

Personal Information

Contact

Name Shivansh Rai
E-mail shivanshrai84@gmail.com
Phone +91 7755047792
IRC zeeb on Freenode