AUTOMATED FIREWALL TESTING

KRISTOF PROVOST
WHO AM I?

- Kristof Provost
- kp@FreeBSD.org
- pf (in FreeBSD) maintainer
- Embedded Linux projects
- EuroBSDCon foundation board member
- Not for sale
  - For rent
    - reasonable rates
WHAT'S THIS PF THING?

PF

- Packet Filter
- Imported from OpenBSD
  - Yes, a while ago
- Shiny things in FreeBSD that are not in OpenBSD
  - vnet
  - multi-core capable
    - Faster!
    - (er)!
WHY AUTOMATED TESTING?

- Make sure things actually work
- Convenient test case
- Prevent regressions
- Quick sanity check when making changes
BAD THINGS TO HAPPEN TO GOOD CODE

REGRESSIONS

- IPv6 fragment handling
  - IPv6 fast path code broke it
  - Took ~9 months to discover and fix
- IPv6 fragments, again
  - Tests found it immediately
  - two weeks between introduction and fix
  - Heisenbug. Went away during DTracing
```c
#include <netinet/in6.h>

int frag6_input(struct mbuf **mp, int *offp, int proto)
{
    /* ... (9 lines) */
    uint32_t hash, hashkey[sizeof(struct in6_addr) * 2 + 1], *hashkeyp;

    /* ... (78 lines) */

    hashkeyp = hashkey;
    memcpy(hashkeyp, &ip6->ip6_src, sizeof(struct in6_addr));
    hashkeyp += sizeof(struct in6_addr) / sizeof(*hashkeyp);
    memcpy(hashkeyp, &ip6->ip6_dst, sizeof(struct in6_addr));
    hashkeyp += sizeof(struct in6_addr) / sizeof(*hashkeyp);
    *hashkeyp = ip6f->ip6f_ident;
    hash = jenkins_hash32(hashkey, nitems(hashkey), V_ip6q_hashseed);
    hash &= IP6REASS_HMASK;
    head = IP6Q_HEAD(hash);
    IP6Q_LOCK(hash);

    /* ... */
}
```
I FIXED A THING!

diff --git a/sys/netinet6/frag6.c b/sys/netinet6/frag6.c
index 0f30801540a..bbdbf448f7c 100644
--- a/sys/netinet6/frag6.c
+++ b/sys/netinet6/frag6.c
@@ -218,7 +218,9 @@
     /* must be larger than u_int16_t */
-    uint32_t hash, hashkey[sizeof(struct in6_addr) * 2 + 1], *hashkeyp;
+    uint32_t hashkey[(sizeof(struct in6_addr) * 2 +
+    sizeof(ip6f->ip6f_ident)) / sizeof(uint32_t)];
+    uint32_t hash, *hashkeyp;
     struct ifnet *dstifp;
     u_int8_t ecn, ecn0;

#ifdef RSS
WHAT DO WE GET OUT OF IT?

OBJECTIVES

- Easy to write
- Easy for everyone to run
- Fast to run
- Integrate with ATF / ci.freebsd.org
Test Result

2 failures (+0), 47 skipped (+3)

All Failed Tests

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Duration</th>
<th>Fail (diff)</th>
<th>Skip (diff)</th>
<th>Pass (diff)</th>
<th>Total (diff)</th>
</tr>
</thead>
<tbody>
<tr>
<td>lib.msun.cbrt.cbrt.powl</td>
<td>4 ms</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>lib.msun.trig.test.reduction</td>
<td>4 ms</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

All Tests

<table>
<thead>
<tr>
<th>Package</th>
<th>Duration</th>
<th>Fail (diff)</th>
<th>Skip (diff)</th>
<th>Pass (diff)</th>
<th>Total (diff)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin.cat</td>
<td>0.21 sec</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>bin.chflags</td>
<td>97 ms</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>bin.chmod</td>
<td>0.47 sec</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>bin.date</td>
<td>1.8 sec</td>
<td>0</td>
<td>0</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>bin.dd</td>
<td>0.26 sec</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>bin.echo</td>
<td>63 ms</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>bin.expr</td>
<td>0.47 sec</td>
<td>0</td>
<td>0</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>bin.in</td>
<td>0.58 sec</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>bin.la</td>
<td>8.9 sec</td>
<td>0</td>
<td>0</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>bin.mkdir</td>
<td>84 ms</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>bin.mv</td>
<td>0.58 sec</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bin.pax</td>
<td>0.12 sec</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bin.pkill</td>
<td>55 sec</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>bin.pwalt</td>
<td>49 sec</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>bin.rm</td>
<td>50 ms</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bin.rmdir</td>
<td>61 ms</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>bin.sh.builts</td>
<td>6.5 sec</td>
<td>0</td>
<td>0</td>
<td>169</td>
<td>169</td>
</tr>
</tbody>
</table>
Take One: Hardware

- Send packets from A to B, check replies
  - Server / switch / server
- But what if we want to forward?
  - Server / switch / server / switch / server
- What if we want to test pfsync or carp?
  - server / switch / server + server / switch / server
ISSUES WITH TAKE ONE

- What if we block all traffic?
  - Serial lines?
- What pf or FreeBSD version on all systems?
  - Netboot?
- Panics?
- What about even more complex setups?
- Where does all this hardware live?
- How do other people write tests?
  - Standardised hardware?
TAKE TWO: VIRTUAL HARDWARE

- bhyve!
- Approach taken in GSoC 2017
ISSUES WITH TAKE TWO

- What if we block all traffic?
  - Emulated serial port
- Nested bhyve ... (ci.freebsd.org)
- Really annoying to build VM during test run
- Panics? Possible, but still annoying
- Slow to run
TAKE THREE: VNET

- Virtual network stack
  - Associated with jail
  - Enabled by default in 12.0
  - pf supports this (as of 12.0)
OKAY, SO HOW DO I START A JAIL WITH ITS OWN STACK? I BET IT’S HARD. IT’S HARD ISN’T IT?

- `sudo jail -c name=alcatraz vnet persist`
WHAT? NO NETWORK? I BET THAT'S HARD!

- `sudo ifconfig epair create`
  - `epair0a / epair0b`
- `sudo ifconfig epair0a 192.0.2.1/24 up`
- `sudo jail -c name=alcatraz vnet persist vnet.interface=epair0b`
- `sudo jexec ifconfig epair0b 192.0.2.2/24 up`
- `ping -c 1 192.0.2.2`
```bash
# $FreeBSD$

. $(atf_get_srcdir)/utils.subr

atf_test_case "v4" "cleanup"
v4_head()
{
  atf_set descr 'Basic pass/block test for IPv4'
  atf_set require.user root
}
```
v4_body()
{
    pft_init

    epair=$({pft_mkepair})
    ifconfig ${epair}a 192.0.2.1/24 up

    # Set up a simple jail with one interface
    pft_mkjail alcatraz ${epair}b
    jexec alcatraz ifconfig ${epair}b 192.0.2.2/24 up

    # Trivial ping to the jail, without pf
    atf_check -s exit:0 -o ignore ping -c 1 -t 1 192.0.2.2

    # pf without policy will let us ping
    jexec alcatraz pfctl -e
    atf_check -s exit:0 -o ignore ping -c 1 -t 1 192.0.2.2

    # Block everything
    pft_set_rules alcatraz "block in"
    atf_check -s exit:2 -o ignore ping -c 1 -t 1 192.0.2.2
}
v4_cleanup()
{
    pft_cleanup
}

atf_init_test_cases()
{
    atf_add_test_case "v4"
}
% sudo kyua test pass_block:v4

pass_block:v4  ->  passed [1.200s]

Results file id is usr_tests_sys_netpfil_pf.20190106-081724-193657

Results saved to /root/.kyua/store/results.usr_tests_sys_netpfil_pf.20190106-081724-193657.db

1/1 passed (0 failed)
```bash
basic_body()
{
    pfsync_init

    epair_sync=$(pft_mkepair)
    epair_one=$(pft_mkepair)
    epair_two=$(pft_mkepair)

    pft_mkjail one ${epair_one}a ${epair_sync}a
    pft_mkjail two ${epair_two}a ${epair_sync}b

    # pfsync interface
    jexec one ifconfig ${epair_sync}a 192.0.2.1/24 up
    jexec one ifconfig ${epair_one}a 198.51.100.1/24 up
    jexec one ifconfig pfsync0 \ syncdev ${epair_sync}a \ maxupd 1 \ up
    jexec two ifconfig ${epair_two}a 198.51.100.2/24 up
    jexec two ifconfig ${epair_sync}b 192.0.2.2/24 up
    jexec two ifconfig pfsync0 \ syncdev ${epair_sync}b \ maxupd 1 \ up
```
# Enable pf!
```
jexec one pfctl -e
pft_set_rules one \"set skip on ${epair_sync}a\" \n  "pass keep state"
jexec two pfctl -e
pft_set_rules two \"set skip on ${epair_sync}b\" \n  "pass keep state"
```

```
ifconfig ${epair_one}b 198.51.100.254/24 up
```

```
ping -c 1 -S 198.51.100.254 198.51.100.1
```

# Give pfsync time to do its thing
```
sleep 2
```

```
if ! jexec two pfctl -s states | grep icmp | grep 198.51.100.1 | \n  grep 198.51.100.2 ; then
  atf_fail "state not found on synced host"
fi
```
WHERE TO FIND THE TESTS

- **Source**
  - /usr/src/tests/sys/netpfil/pf

- **Installed**
  - /usr/tests/sys/netpfil/pf
DO IT YOURSELF TESTING

HOW DO I RUN TESTS?

- pkg install kyua scapy
- kldload pfsync
- cd /usr/tests/sys/netpfil
- kyua test
MORE INFORMATION

- FreeBSD journal
  - March/April 2019
  - https://www.freebsd foundation.org/journal/
SERIOUSLY, WRITE TESTS. TESTS ARE GOOD.

Me. Just now.
SERIOUSLY, WHY THOUGH?

WHAT’S IN IT FOR YOU?

- Prototype setups
- Prevent your use case from breaking
- Make it easy for me to fix your bug
  - Seriously. I’m lazy. Make it easy
  - Often reproducing is more than half of the actual work
    - Assuming I even understand your setup
  - With a good test it’s often easier to fix than to review a patch
    - I’d have to write the test anyway. Do it for me
- Money also motivates me
"I CAN’T BE FIRST!"

**OTHER VNET TESTS**

- netipsec
  - Olivier was tired of IPSec being broken
  - Now
    - there are tests
    - IPSec isn’t broken
    - If someone does break it, Li-Wen will shout[*] at them

[*] WELL… POLITELY ASK THEM TO FIX IT
QUESTIONS?
ONE MORE?
NAT: OBSCURE NAT PROBLEM (1/2)

```bash
exhaust_body()
{
    pft_init
    epair_nat=$(vnet_mkepair)
    epair_echo=$(vnet_mkepair)

    vnet_mkjail nat ${epair_nat}b ${epair_echo}a
    vnet_mkjail echo ${epair_echo}b

    ifconfig ${epair_nat}a 192.0.2.2/24 up
    route add -net 198.51.100.0/24 192.0.2.1

    jexec nat ifconfig ${epair_nat}b 192.0.2.1/24 up
    jexec nat ifconfig ${epair_echo}a 198.51.100.1/24 up
    jexec nat sysctl net.inet.ip.forwarding=1

    jexec echo ifconfig ${epair_echo}b 198.51.100.2/24 up
    jexec echo /usr/sbin/inetd $(atf_get_srcdir)/echo_inetd.conf

    # Enable pf!
    jexec nat pfctl -e
    pft_set_rules nat \
        "nat pass on ${epair_echo}a inet from 192.0.2.0/24 to any 
        -> (${epair_echo}a) port 30000:30001 sticky-address"
```
# Sanity check
```bash
atf_check -s exit:0 -o ignore ping -c 3 198.51.100.2
```
```bash
echo "foo" | nc -N 198.51.100.2 7
```
```bash
echo "foo" | nc -N 198.51.100.2 7
```
```bash
# This one will fail, but that's expected
echo "foo" | nc -N 198.51.100.2 7 &
```
```bash
sleep 1
```
```bash
# If the kernel is stuck in pf_get_sport() this will not succeed either.
timeout 2 jexec nat pfctl -sa
if [ $? -eq 124 ]; then
    # Timed out
    atf_fail "pfctl timeout"
fi
```
```bash
}
```