Through the Wire: Measurement and Improvement of a software based IPsec implementation

George Neville-Neil Jim Thompson

BSDCan 2016

Benchmarks are Hard

- What do we measure?
- How do we measure it?
- How do we verify our measurements?
- Can our measurement be repeated?
- Can our measurement be replicated?
- Is our measurement relevant?
- How do we generate a workload?
- Does our measurement technology disturb the measurement?
    - Heisentesting

Network Benchmarks are Harder

- Asynchrony
- Best effort delivery
- Lack of open source test tools
- Control of distributed systems

Modern Hardware

- 10Gbps is 14.8 million 64 byte packets per second
- 67.5ns per packet or 200cycles at 3GHz
- Cache miss is 32ns
- Multi-core
- Multi-queue
- Lining it all up

Packet Processing Times

length (bytes)	rate (Mpps)	cycle budget @ 3GHz
64	14.88	201
128	8.44	355
256	4.52	663
512	2.34	1282
1024	1.19	2521
1280	0.961	3121
1518	0.861	3484

Test Automation: Conductor

- Set of Python libraries
- *Conductor* and 1, or more, *Players*
- Four Phases

Startup	Set up system, load drivers, set routes, etc.
Run	Execute the test
Collect	Retrieve log files and output
Reset	Return system to original state

Conductor Config

```
1   # Master config file to run an iperf test WITHOUT PF enabled.
2   [Test]
3   trials: 1
4
5   [Clients]
6   # Sender
7   client1: source.cfg
8   # DUT
9   client2: dut.cfg
10  # Receiver
11  client3: sink.cfg
```

Player Config

```
1   [Master]
2   player: 192.168.5.81
3   conductor: 192.168.5.1
4   cmdport: 6970
5   resultsport: 6971
6
7   [Startup]
8   step1:ifconfig ix0 172.16.0.2/24
9   step2:ifconfig ix1 172.16.1.2/24
10  step3:ping -c 3 172.16.0.1
11  step4:ping -c 3 172.16.1.3
12
13  [Run]
14  step1:echo "running"
15  step2:pmcstat -O /mnt/memdisk/pktgen-instruction-retired.pmc -S instruction-retired -l 25
16
17  [Collect]
18  step1:echo "collecting"
19  step2:mkdir /tmp/results
20  step3:cp -f /mnt/memdisk/pktgen-instruction-retired.pmc /tmp/results/
21  step4:pmcstat -R /tmp/results/pktgen-instruction-retired.pmc -G \
22            /tmp/results/pktgen-instruction-retired.graph
23  step5:pmcstat -R /tmp/results/pktgen-instruction-retired.pmc -D /tm/results -g
24  step6:pmcannotate /tmp/results/pktgen-instruction-retired.pmc \
25              /boot/kernel/kernel > /tmp/results/pktgen-instruction-retired.ann
26
27  [Reset]
28  step1:echo "system reset: goodbye"
```

Zopkio

- Developed by LinkedIn
- Meant for Distributed Systems Testing
- Imported into ports by Marcelo Aruajo
- benchmarks/py-zopkio
- More complicated to set up

Host to Host Baseline Measurement

iperf TCP based test

pktgen Packet based test using `netmap(4)`

Baseline TCP Measurement

```
0.00-1.00   sec 1.09 GBytes 9.41 Gbits/sec
1.00-2.00   sec 1.10 GBytes 9.41 Gbits/sec
2.00-3.00   sec 1.10 GBytes 9.41 Gbits/sec
3.00-4.00   sec 1.10 GBytes 9.41 Gbits/sec
4.00-5.00   sec 1.10 GBytes 9.41 Gbits/sec
5.00-6.00   sec 1.10 GBytes 9.42 Gbits/sec
6.00-7.00   sec 1.10 GBytes 9.41 Gbits/sec
7.00-8.00   sec 1.10 GBytes 9.41 Gbits/sec
8.00-9.00   sec 1.10 GBytes 9.41 Gbits/sec
9.00-10.00 sec 1.10 GBytes 9.41 Gbits/sec
```

Baseline pkt-gen Measurement

- Source

```
827.257743 main_thread [1512] 14697768 pps
828.259812 main_thread [1512] 14668997 pps
829.261742 main_thread [1512] 14695277 pps
830.263743 main_thread [1512] 14685547 pps
```

- Sink

```
866.466039 main_thread [1512] 11943109 pps
867.468024 main_thread [1512] 11946111 pps
868.469126 main_thread [1512] 11942020 pps
869.471027 main_thread [1512] 11939957 pps
```

Baseline Discussion

- TCP uses full sized packets
- pkt-gen uses minimum sized (64 byte) packets
- The DUT cannot quite keep up

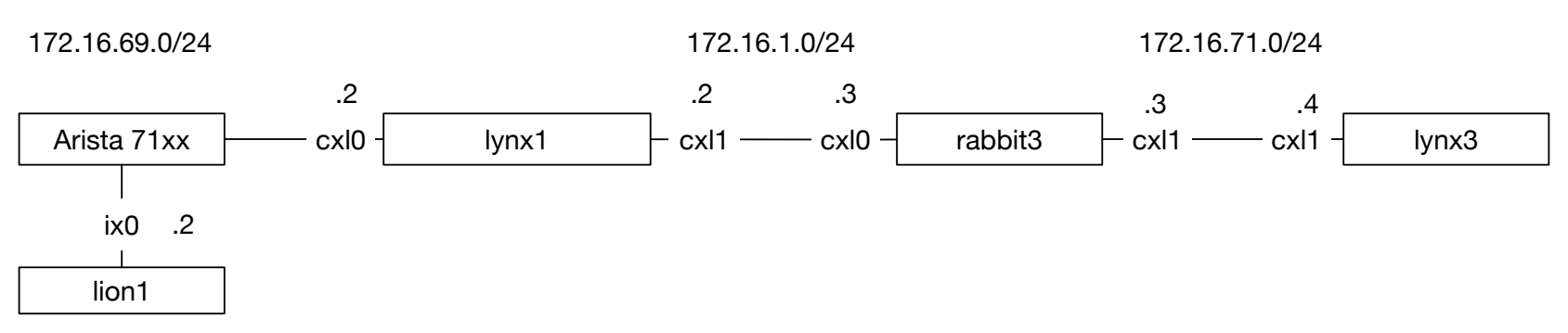
IPsec and its Algorithms

- Encryption is computationally expensive
- Offloaded co-processors
- On chip instructions `AES-NI`

Measurement Methods

- Four host setup
  - Source, VPN Left, VPN Right, Sink
- iperf3 using TCP
- Conductor sets up the tests
- 10 rounds of 10 seconds each

Lab Setup


172.16.69.0/24
172.16.1.0/24
172.16.71.0/24
.2
.2
.3
.3
.4
Arista 71xx
cxl0
lynx1
cxl1
cxl0
rabbit3
cxl1
cxl1
lynx3
ix0
.2
lion1

Hardware Used

Host	Purpose	CPU	Cores	NIC
lynx3	Packet Source	E5-2680 v2 2.80GHz	10	T5
rabbit3	VPN Right (Ingress)	E5-2637 v2 3.50GHz	4	T5
lynx1	VPN Left (Egress)	E5-2680 v2 2.80GHz	10	T5
lion1	Packet Sink	X5365 3.00GHz	8	Intel

Baseline

```
1  static void
2  null_encrypt(caddr_t key, u_int8_t *blk)
3  {
4  }
5
6  static void
7  null_decrypt(caddr_t key, u_int8_t *blk)
8  {
9  }
```

- Using NULL methods
- No authentication or encryption
- No TCP offload on the NIC cards
- Who needs NULL encryption?

Baseline Results

Min	Max	Median	Avg	Stddev
4.67	4.88	4.75	4.761	0.070781353

Baseline Investigation

- Use of hwpmc(4) to answer questions.
- Where does the time go?
- Computationally expensive?
- Are we abusing the caches?

A note about hwpmc(4)

- Has a probe effect
- Measurement skid
- Need independent verification
- Could use better visualization tools

Baseline Investigation Results

%	self	descendents	function
64.5	15595.00	751147.08	ip_input [1]
	10886.00	627426.78	ip_forward [5]
	125.00	88506.65	esp4_input [16]
	8284.07	664.33	__rw_rlock [22]
	7539.97	0.00	_rw_runlock_cookie [32]
	2793.00	1522.50	ip_ipsec_filtertunnel [109]
	3388.48	0.00	key_havesp [95]
	6.39	0.78	m_dup_pkthdr [75]
	3.14	0.00	ip_ipsec_fwd [260]

NULL Encryption Flat Profile

%	cumulative	self	calls	function
8.6	102273.00	102273.00	7407	sched_idletd [11]
5.4	165999.00	63726.00	39263	__rw_rlock [22]
4.6	221120.00	55121.00	10114	bcopy [28]
4.5	274695.00	53575.00	12600	bzero [29]
4.5	328199.00	53504.00	44697	uma_zfree_arg <cycle 5> [21]
4.2	378037.00	49838.00	20588	uma_zalloc_arg <cycle 5> [15]
3.8	423751.00	45714.00	10798	_rw_runlock_cookie [32]

AES-CBC with SHA1

- Tunnel Mode
- Encryption with SHA1 authentication

HW Suppt	Min	Max	Median	Avg	Stddev
N	393	420	400	403.59	8.55
Y	303	418	400	400	11

AES-GCM

- Tunnel Mode
- Both encryption and authentication
- Results for 128 bit keys with and without hardware support

Crypto	Min	Max	Median	Avg	Stddev
Soft	247	281	276	269.9	12.844
Hard	1820	1880	1840	1843.0	0.022

AES-GCM Instructions Retired

%	self	descendents	function
48.0	3394.00	481170.96	swcr_process [6]
	19972.00	239483.38	aes_icm_crypt [17]
	684.00	59949.12	AES_GMAC_Final [22]
	30131.00	22363.76	AES_GMAC_Update [24]
	13797.99	22752.61	m_copydata [23]
	205.00	25886.93	esp_input_cb [28]
	16239.66	7433.05	m_copyback [26]
	4669.58	0.00	crypto_copyback [44]
	4225.99	0.00	bzero [25]
	4045.87	0.00	crypto_copydata [49]
	3748.29	0.00	bcopy [21]
	1897.52	330.74	__rw_rlock [47]
	129.00	1589.38	aes_gcm_reinit [67]
	1499.16	0.00	_rw_runlock_cookie [51]
	136.94	0.00	timingsafe_bcmp [152]

AES-GCM Instructions Retired Flat Profile

%	cumulative	self	calls	function
30.7	309697.00	309697.00	0	gf128_mul [15]
28.9	601037.00	291340.00	4721	rijndaelEncrypt [16]
9.7	699215.00	98178.00	2276	sched_idletd [19]
6.0	759854.00	60639.00	1456	bcopy [21]
4.4	803854.00	44000.00	885	bzero [25]
3.0	833985.00	30131.00	477	AES_GMAC_Update [24]
2.2	855933.00	21948.00	1193	m_copydata [23]
2.1	877225.00	21292.00	354	m_copyback [26]
2.0	897197.00	19972.00	3911	aes_icm_crypt [17]

Hardware Assisted AES-GCM Instructions Retired Flat Profile

%	cumulative	self	calls	function
20.3	129263.00	129263.00	15237	sched_idletd [12]
6.0	167470.00	38207.00	31881	bcopy [20]
3.9	192108.00	24638.00	45240	uma_zfree_arg <cycle 3> [22]
3.8	216589.00	24481.00	41084	uma_zalloc_arg <cycle 3> [18]
3.5	238762.00	22173.00	21117	bzero [29]
3.4	260335.00	21573.00	18845	__rw_rlock [28]
2.3	274920.00	14585.00	12399	_rw_runlock_cookie [43]

Discussion

- Math on the CPU takes a long time
- CPU is freed for other work with hardware assist
- Locking and memory management dominate the remaining measurements
- Confirms our suspicions from the NULL measurements
- Packet at a time has high overhead

Tune in Next Time...

- OCF vs. IPSEC
- TCP Testing
- Using Dummynet for what it was meant for
- Congestion Control Comparisons

An Ongoing Longitudinal Study

- Conducted continuously
- Reported several times per year
- Covering more subsystems
- AsiaBSDCon: March, 2015 ✓
- BSDCan: June, 2015 ✓
- EuroBSDCon: October 2015 ✓
- AsiaBSDCon: March 2016 ✓
- BSDCan: June 2016 ✓
- EuroBSD: 2016 ???
- Get it here: `http://github.com/gvnn3/netperf`