ACCELERATE SNORT WITH HYPERSCAN

Xiang Wang
Software Engineer
xiang.w.wang@intel.com

Legal Disclaimer

General Disclaimer:

© Copyright 2015 Intel Corporation. All rights reserved. Intel, the Intel logo, Intel Inside, the Intel Inside logo, Intel. Experience What's Inside are trademarks of Intel. Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

Technology Disclaimer:

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Performance Disclaimers:

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Agenda

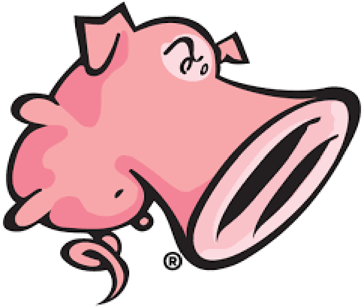
- Snort Overview
- Integrating Snort with Hyperscan
- Experiments and Demo

SNORT OVERVIEW

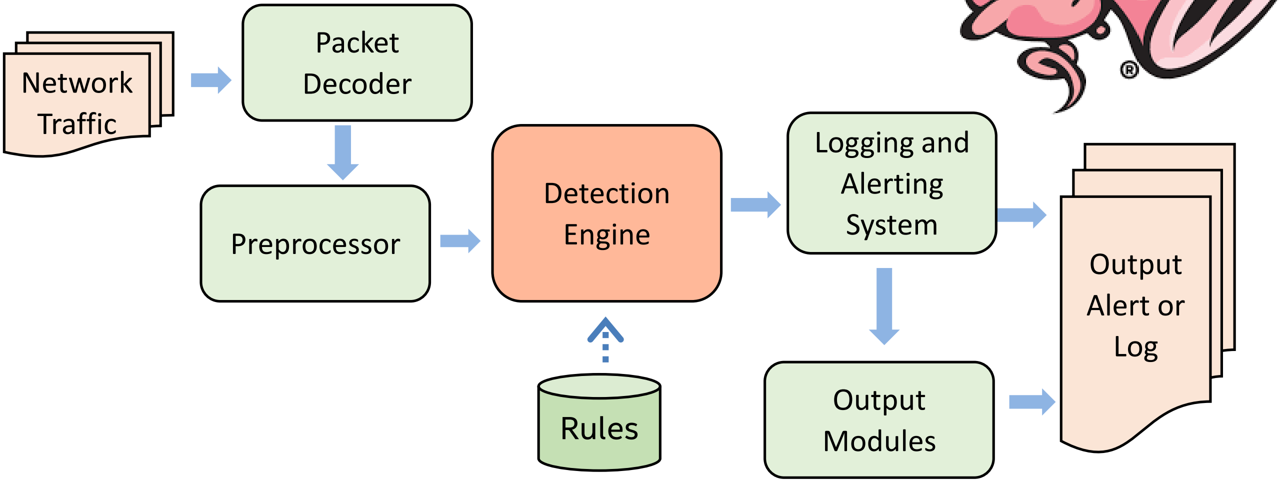
Snort Architecture

Open Source Intrusion {Detection, Prevention} System

- Cisco (previously Sourcefire) owns GPL
- Most widely deployed IPS/IDS in the industry
- Public Snort VRT rules targeting at hacking activities, intrusion attempts, malware and vulnerabilities, etc
- Single-threaded architecture in Snort 2.x (Multi-thread support in the coming Snort 3.0)
- First beta release of Snort 3.0 is expected at the mid of 2017



Snort Introduction



Network Traffic

Packet Decoder

Preprocessor

Detection Engine

Rules

Logging and Alerting System

Output Modules

Output Alert or Log

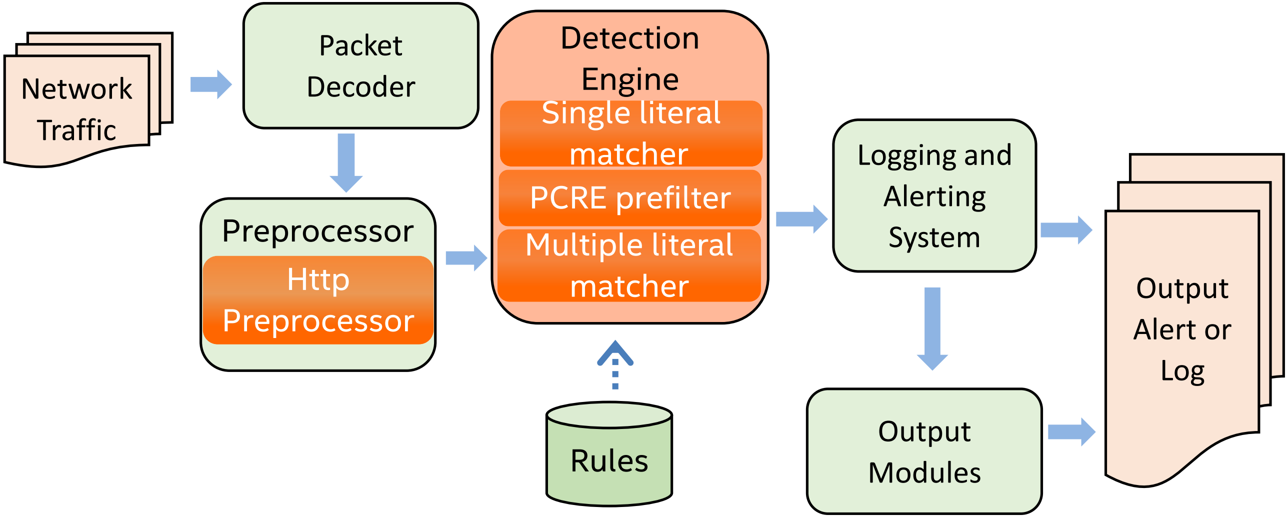
INTEGRATING SNORT WITH HYPERSCAN

Integration Overview

Two integrations: integration into Snort 2.9 series and Snort 3 aka Snort++

- Snort 2.9 integration (Intel)
  - Uses Hyperscan as multiple literal matcher aka “MPSE”
  - Uses Hyperscan as single literal matcher (!!)
  - Uses Hyperscan as regex matcher
  - Uses Hyperscan in http preprocessor
  - Not upstreamed – we ship patch at *01.org/hyperscan*
- Snort 3 integration (Cisco)
  - Experimental – allows explicit regular expressions in the ‘multiple matcher’

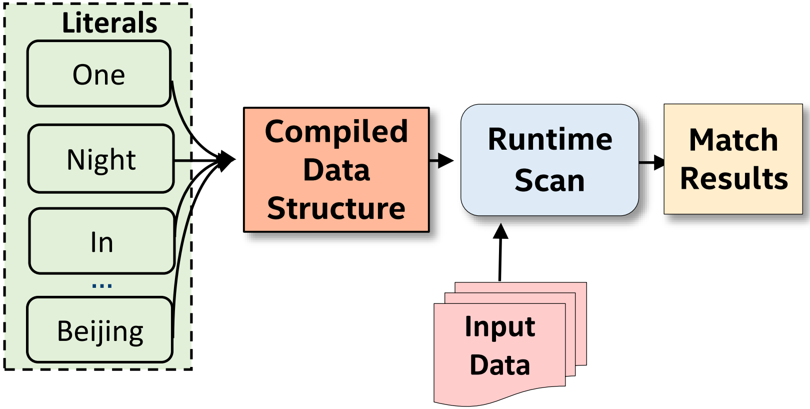
Accelerated Snort



Detection Engine Integration

Multiple literal matching

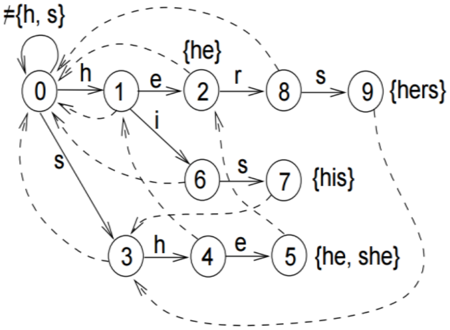
- Fast pattern matching (for all rules):
  - Core component in detection engine
  - Only evaluate rules if the content is found in the payload
  - Significantly reduce the number of rules to evaluate and thus better performance



Detection Engine Integration

Multiple literal matching:

- Aho-Corasick algorithm *(default)*



- Replace AC with Hyperscan

  Mainly use *large scale literal matcher* in Hyperscan

  - Bucketed super-character shift-or matcher as front end
  - Hashing and final confirm as back end
  - Based on Intel instructions: SSE + BMI
  - 500 literals: ~10Gbps, 5000 literals: ~4Gbps

Detection Engine Integration

Individual rule matching:

- Single literal matching:
  - Boyer-Moore algorithm (*default*)
  - Use Hyperscan *SIMD based single literal matcher*
- Regular expression matching:
  - PCRE (*default*)
  - Use Hyperscan's *prefilter* support to handle patterns that Hyperscan doesn't natively support
  - Scan with Hyperscan first, confirm with PCRE if Hyperscan has matched
  - Significant win when PCRE will backtrack, or when literal guard is weak

Detection Engine Integration

Multiple literal matching code snippet:

```
typedef struct _HyperscanContext {
    hs_scratch_t *scratch;
} HyperscanContext;

typedef struct _HyperscanPm {
    hs_database_t *db;
    HyperscanContext *ctx;
    HyperscanPattern *patterns;
    …
} HyperscanPm;
```

```
typedef struct
HyperscanCallbackContext_ {
    const HyperscanPm *pm;
    void *data;
    int (*match)(void *id, …);
    int num_matches;
} HyperscanCallbackContext;
```

Detection Engine Integration

```
static int
HyperscanBuild(HyperscanContext
                    *ctx,
                    HyperscanPm *pm) {
  hs_compile_ext_multi(patterns,
                          flags, ids,
                          ext,
                          num_patterns,
                          HS_MODE_BLOCK,
                          NULL,
                          &(pm->db),
                          &compile_error);

   hs_alloc_scratch(pm->db,
                       &pm->ctx->scratch);
}
```

```
int HyperscanSearch(HyperscanPm *pm, …) {
   HyperscanCallbackContext ctx;
   hs_scan(pm->db, (const char *)t, tlen, 0,
               pm->ctx->scratch, onMatch, &ctx);
   return ctx.num_matches;
}
```

```
static void HyperscanCleanup(int unused,
                                          void *data) {
   hs_free_scratch(contentScratch);
   contentScratch = NULL;
}
void HyperscanFree(HyperscanPm *pm) {
         hs_free_database(pm->db);
}
```

EXPERIMENTS AND DEMO

Performance

CPU: Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz

Hyper-threading: disabled

Turbo Boost: enabled (Max 3.6GHz)

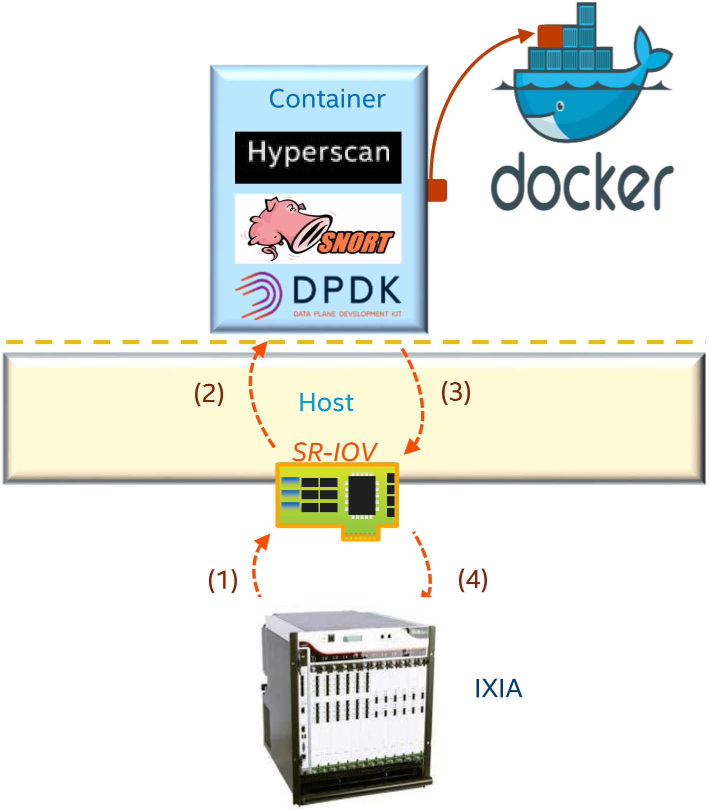
NIC: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+

HugePage: 4 * 1G

Snort/DAQ/Rules: Snort 2.9.8.3, daq 2.0.6, snortrules-snapshot-2983

DPDK/Hyperscan: DPDK 16.07, Hyperscan 4.3.1

BreakingPoint: Release 8.0.1



Performance

Average Throughput (Mbps)

Rules	Snort/DPDK/Hyperscan	Vanilla Snort	Performance Improvement
0 rule & direct forwarding	41,625.4535	4,294.8543	*9.69x*
0 rule	4,346.5399	1,720.2997	*2.53x*
8 pass rules	1,823.7223	841.8590	*2.17x*
VRT rule package	772.5730	91.4068	*8.45x*

Traffic containing HTTP and peer-to-peer file sharing

Performance

Average Throughput (Mbps)

Rules	Snort/DPDK/Hyperscan	Vanilla Snort	Performance Improvement
0 rule & direct forwarding	9,897.0288	1,164.4064	*8.50x*
0 rule	1,918.4513	656.8847	*2.92x*
8 pass rules	666.9908	377.4932	*1.77x*
VRT rule package	314.3337	118.7311	*2.65x*

Traffic representing an enterprise network

CONCLUSIONS AND CALL TO ACTION

Conclusion

- Hyperscan
  - Contributes substantial speedups to Snort
  - Solid and mature for pattern matching intensive systems (IDS/IPS/FW…)
  - Delivers strong packet processing capability together with DPDK
- Call to action:
  - Stay in touch with us! Help us to improve Hyperscan for network solutions.
  - Welcome ideas on projects in various fields (text analytics, bioinformatics, etc)