HYPERSCAN IN SURICATA

Chi Xu

Hyperscan Software Engineer

chi.xu@intel.com

Legal Disclaimer

General Disclaimer:

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 are provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

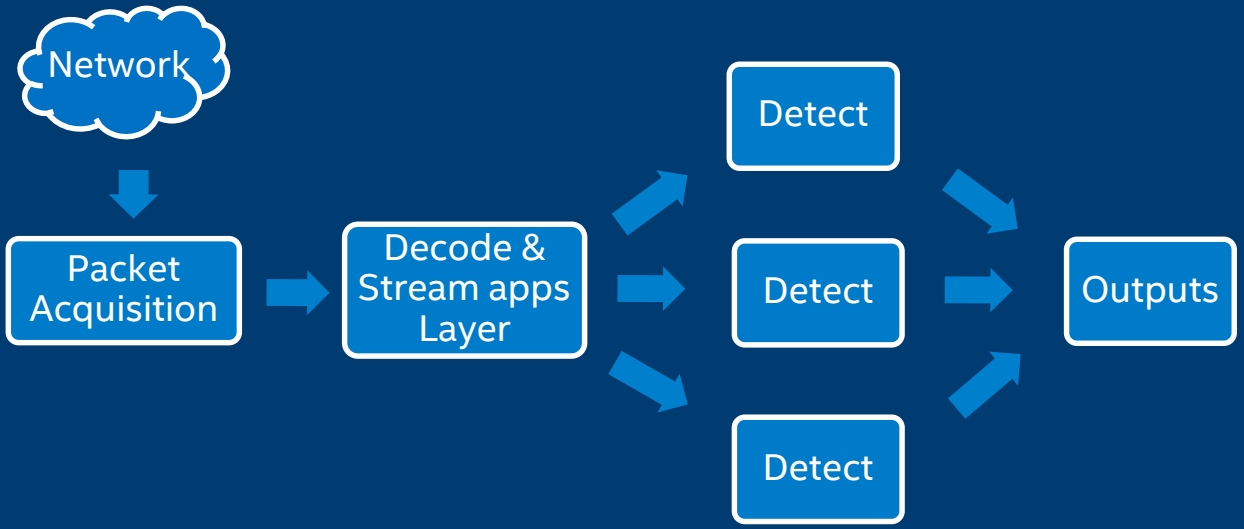
Welcome and Goals

- Goals:
  - Show what we've done with Suricata
  - Show where we're hitting the limits
  - Show how to get better for your models

Suricata

- Suricata is a GPL-licensed Snort competitor with a similar design, rule format, run by the OISF and also widely used
- Fully support Snort rules
- Multi-threaded already, unlike Snort 2.x
- Developed in the open, using Github

Suricata Block Diagram



Suricata Integration

Our Suricata+Hyperscan patch has patched several places in scanning:

Suricata (Since 3.1 release) supports:

- Single literal match
- Multiple literal match
  - Default option on supported platforms

Experimental work only :

- Single regular expression match

Suricata Integration: Single Literal Match

Noodle: fast single literal matcher

- Tuned SIMD match on Intel – >100+Gbps typical
- Examine 1-2 characters, do the PCMPEQB/shift/and SIMD tricks
  - Gets faster on AVX2, AVX 512
- Use Hyperscan opportunistically
  - Just literals in this pass

Noodle: Single Literal Match Engine

To match literal /Hyperscan/, build two 16-byte masks

H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y

For input data "*MATCHmyHyperscan*" at run-time, do PCMPEQB

0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0

LSHIFT and AND

0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Suricata Integration: Multiple Literal Match

- Large scale literal match a ‘staple’ of Hyperscan
  - Three algorithms
    - FDR: ‘bucketed super-character shift-or’ (default)
    - Teddy: “SIMD-based bucketed matcher’ (2-72 literals)
    - Noodle (SPM): fast SIMD matcher for 1 literal
- Just literals in this pass?
  - No: we do additional constructs like /^A{10,50}xyz/s
    - “Anchored” patterns
    - Avoid match “floods”

Suricata Integration: Single Regex

Simulate behavior of backtracking regular expressions adequately

- Implement *all* regular expressions, including back-references and zero-width asserts, etc.

Standard escape hatch: Pre-filtering (false positive but not false negative)

- HS_MODE_PREFILTER: replace unsupported constructs with weaker substitutes
  - Example \foo(\d)+bar\1baz\ -> \foo(\d)+bar(\d)+baz\

Suricata Integration – Performance results

Machine under test: Intel® Core i7 6700K CPU @ 4.0 GHz

Software versions:

- Hyperscan 4.3.1
- Suricata 3.1

The following disclaimer applies to the results presented in the next slides:

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/performance

Suricata Integration – Performance results

Rule sets

- Emerging Threats public set (“emerging-all-20161102.rules”)
- ET Pro set (“etpro-all-20161102.rules”)

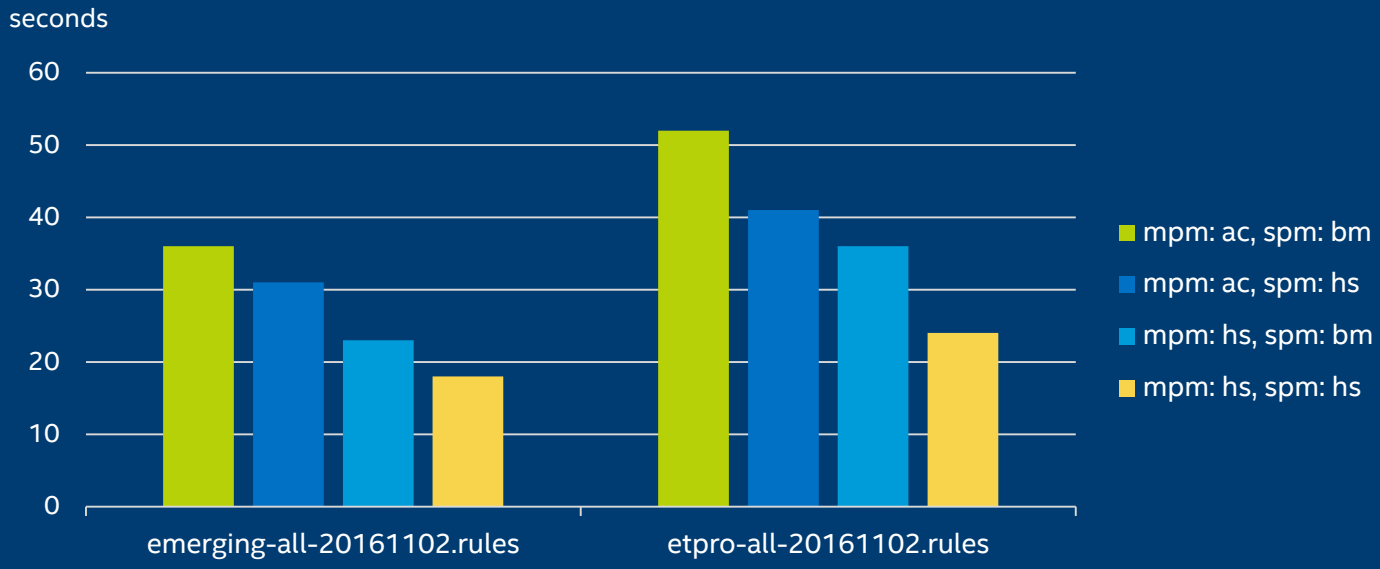
Input PCAPs: Alexa Top 100 browsing PCAP file

- Measurement is “all processing time” (not just scan)
  - But it’s not true end-to-end (not a network measurement)
  - If we compared head to head against Aho-Corasick the Hyperscan advantage would be bigger

Hyperscan vs default Aho-Corasick

All Hyperscan vs AC+BM: Emerging Threats: 1.95x, ET Pro: 2.15x speedup

Measured in elapsed time in seconds for our PCAP file



Performance: Regular Expression “Acceleration”

Rules	PCRE elapsed time	Hyperscan elapsed time
emerging-all-20161102.rules	18.3	18.5
etpro-all-20161102.rules	24.1	24.3

No benefit from Hyperscan overs libpcre for 1-at-a-time-regex

Why No Regex Speedup?

Hyperscan is big and not optimized for single pattern scans

- Self-interference and interference with other Suricata code

What we can do:

- Cherry-pick some new cases?
- Move regex portion of workload to more appropriate place (multiple matching)
- Use Hyperscan in streaming mode

Conclusions

Suricata integration with Hyperscan for MPM and SPM

Hyperscan doubles performance and has less memory cost

- **Call to action**: try Hyperscan if you haven't already

Single regex matching is an awkward fit
Plenty of low-hanging fruit in Suricata scanning model

- **Call to action**: fix model to use streaming & multi-matching

Questions?