

# Increasing Snort® Performance By Replacing Pattern Matching Algorithms

## Intel® QuickAssist Technology reduces cost and increases scalability for network and security appliances

Ubiquitous connectivity and increasing network data rates are driving an ever increasing computational demand for processing workloads such as pattern matching, cryptography, and data compression. Intel® QuickAssist Technology enables the use of content processing accelerators to address this demand with innovative solutions from multiple vendors. These solutions provide maximum performance while minimizing power consumption, footprint, and cost.

Implementations of Intel QuickAssist Technology are available from Intel today, and new implementations are in development. Meanwhile, Intel has been working with its partners in the Intel QuickAssist Technology Community to encourage the development of innovative accelerator solutions. The Intel QuickAssist Technology Community is a virtual organization of Independent Software Vendors, Independent Hardware Vendors, embedded Original Equipment Manufacturers, and developers, who are committed to simplifying accelerator use on Intel® architecture platforms. More details about the community and about Intel QuickAssist Technology can be found on the portal site, <http://qatc.intel.com>.

Intel QuickAssist Technology defines a unified set of Application Programming Interfaces (APIs) providing consistent conventions and semantics across solutions available from the Intel QuickAssist Technology Community. Implementations of Intel QuickAssist Technology can range from optimized software algorithms to high performance hardware accelerators. For each workload, application developers can write to a single API and deploy the optimal solution for a given platform. This solution brief describes Sourcefire's use of Intel QuickAssist Technology for Pattern Matching in Snort®, an open source network intrusion prevention and detection system (IDS/IPS). A software-based solution was utilized to achieve performance gains over its internal pattern matching algorithms.



## Benefits of Intel® QuickAssist Technology

Some of the key benefits to system developers choosing to deploy accelerators based on Intel QuickAssist Technology are as follows:

**Scalability:** Intel QuickAssist Technology allows a choice of accelerators with different performance characteristics in either a single instantiation or with multiple accelerators. Combined with a range of available processors, a product can scale in general purpose computes as well as content processing capacity.

**Software Cost:** A consistent API that provides a rich and powerful set of features simplifies software reuse across various products over multiple development cycles, thus protecting software investments.

**Compatibility and Interoperability:** Having a community of vendors working to define and implement a consistent set of APIs allows for compatible, interoperable solutions. This increases competition and reduces risk for system developers.

**Portability:** The APIs are designed to be operating system and user/kernel space independent. Memory is allocated by the calling application and provided to the API implementation through the API.

**Performance:** The API supports both synchronous and asynchronous invocation modes. Flexible memory models are supported for data buffers to allow for zero-copy user space implementations.

### Details of Workloads

Intel QuickAssist Technology uses a combination of base and workload specific APIs, such as pattern matching, cryptography, and data compression, which are described in the following.

#### Base API

The base API defines a number of types, structures, and other constructs that are common across all of the services and APIs. The base API defines the discovery and initialization of accelerator instances and the data buffer representations.

An accelerator instance represents a channel between an application and an accelerator. Instance discovery allows enumeration of the instances for a particular workload and the query of individual instances. Vendor information, version number, operational state, and memory model are some of the values that can be obtained through the discovery mechanism. Instance capabilities can also be queried to determine which optional features are supported.

To avoid buffer copying, the majority of operating system network stacks support some form of scatter-gather list. The corresponding representation in the base API is a buffer list, which

consists of an array of flat buffers. Most of the source and destination buffers in the APIs use this format.

The following workloads are comprehended by Intel QuickAssist Technology today.

#### Pattern Matching

The Intel QuickAssist Technology Pattern Matching API allows for individual buffers or data streams to be searched against a database of patterns, which can be a combination of fixed string (or literals) and regular expressions. This service can be used by many “deep packet inspection” applications such as intrusion detection/prevention systems, anti-virus, anti-spam, and other anti-malware applications. The API consists of three major parts, the base API, pattern database management, and pattern matching functions. Some of the key features of the API are:

- A pattern database (PDB) can be further divided into subsets, called pattern groups, to allow the application to specify the relevant patterns, such as the HTTP rules, for a given search request. The API provides a rich set of functionality that encompasses the creation, compilation and activation of PDBs.
- Sessions are used to define the context for pattern match requests. The session defines the relevant search parameters such as the pattern groups and whether the search is stateless or stateful (may cross buffer boundaries). The number of sessions is only bounded by the available memory.
- Multiple pattern matching requests can be made in a single API invocation. For example, an application could request multiple URL searches and a payload search in a single function call.
- The API is inherently asynchronous, and an application may receive the match results either by a callback or by a polling function. The polling function returns the results for all completed match requests, and it supports both blocking and non-blocking options.

#### Cryptography

The Intel QuickAssist Technology Cryptographic API offers a full suite of raw cryptographic primitives, including symmetric and public key cryptography, large number arithmetic, and prime number testing, as well as random number generation. This allows for a range of cryptographic protocols (including IPsec and SSL) and applications (such as VPNs) to be accelerated.

#### Data Compression

The Intel QuickAssist Technology Data Compression API provides access to different compression based implementations supporting several differing algorithms such as deflate, LZS, eLZS, and LZSS. These algorithms support various applications, including HTTP compression, file-based compression, and WAN acceleration.

## Intel® QuickAssist Technology Implementation Example

### Snort®

Snort ([www.snort.org](http://www.snort.org)) is an Open Source intrusion detection/prevention system created in 1998, with the first official release in 1999. Snort performs pattern matching based on a pre-selected group of rules that are used to inspect packets at a deeper level. Pattern matching is often much less expensive in terms of CPU resources when compared to deep rule inspection, so using pattern matching helps to increase overall throughput and reduce the latency of the packets being inspected.

Historically, Snort has made use of many different pattern matchers, ranging from a simple keyword trie, a Wu-Manber algorithm, and different flavors of the Aho-Corasick (AC) algorithm. While a full-state tree of AC provides the best performance, the default pattern matcher in Snort has been a binary NFA implementation of AC (AC-BNFA) since 2006. Since Snort is deployed on a wide range of systems, both large and small, an important consideration is the tradeoff between the amount of memory used for pattern matching and performance.

At initialization time, Snort builds a database of patterns based on the Snort rules being loaded. Most rules have a simple pattern, and Snort uses the most unique pattern from each rule to insert into the database. The most unique pattern is often explicitly identified by the author of the rule, and in cases when it is not, the longest pattern from the rule is used. The database is broken down into groups based on the ports and services listed in the rules. For each group, the patterns are then compiled into a pattern matcher state machine.

Once Snort starts inspecting network traffic, it performs network layer normalization (IP defragmentation, TCP Stream reassembly) and application layer normalization (HTTP, FTP, SMTP, etc.), and selects the group based on the port or service for the given packet. Snort then invokes a multi pattern search engine (MPSE) using the state machine for the port group to select the specific rules for more CPU intensive deep inspection. When a pattern is not present in the payload of the packet, the rules associated with it have no potential to match. After the rules for deep inspection are identified, Snort uses its detection engine to evaluate specific locations of patterns relative to each other, using “regular expressions,” and performs byte comparisons to look for values that could cause overflows (i.e., searching for data that might exploit vulnerability on the receiving host).

## Integration with Intel® QuickAssist Technology Pattern Matching

Snort’s pluggable architecture facilitated the addition of Intel QuickAssist Technology pattern matching mechanisms because the Pattern Matching API largely mirrored the APIs already in-use for the supported algorithms within Snort. The actual changes to Snort were minimal, requiring only an additional 400 lines of executable code. Most of the changes centered around a set of wrapper routines that consolidated the calls to the Pattern Matching APIs into a single module and added configuration options to select the Intel QuickAssist Technology Pattern Matcher.

The Snort development team has a fairly rich suite of regression tests that are used to measure changes in its detection capabilities. Once the code integration was completed and the regression tests run, there were no differences between what Snort detected with the AC-BNFA algorithm and the Pattern Matching implementation within the Intel QuickAssist Technology library.

### Performance Gains

Working with a software version of Intel QuickAssist Technology Pattern Matching, Snort realized some significant performance gains. These gains were noticed in both live traffic and lab performance tests using Snort 2.8.6 integrated with Intel QuickAssist Technology.

### Lab Tests

In a performance test lab, standard tests based on Intrusion Prevention Systems certification tests demonstrated a gain of 8 percent in the pattern matching (MPSE) across different packet sizes when using the Intel QuickAssist Technology library, as compared to Snort’s internal pattern matching algorithm. The traffic for these tests consists of HTTP with responses of varying packet sizes, as well as a mix protocol (HTTP, SMTP, FTP, DNS, etc.) test. For each test, Snort was configured to run first with its internal AC pattern matcher and then with the Intel QuickAssist Technology Pattern Matching API.

The following tables show the performance improvement in the lab tests.

Snort® 2.8.6	MPSE (usec), 128 byte	MPSE (usec), 440 byte	MPSE (usec), 1024 byte
AC	0.79	1.11	2.89
Intel	0.66	0.85	2.63
% Gain	8.35%	7.65%	9.88%

### Lab Test: Latency

## Live Traffic Tests

While the lab tests provide a controlled environment with repeatable tests, a live traffic environment demonstrates a more significant improvement, and one that could be realized by Snort users. In this test, two equivalent hardware systems were placed in parallel so they saw identical traffic. One system was configured to use the native Snort pattern matching algorithm of AC, while the other used the Intel QuickAssist Technology Pattern Matching API.

The breakdown of the live traffic was composed as follows:

- TCP: 77%
  - HTTP: 30%
  - NetBIOS/DCE RPC: 5%
  - Mixed Protocol/High Port: 40+%
- UDP: 22%
- Other IP/ARP: < 1%

The following table shows the performance improvement measured with live traffic tests.

Snort® 2.8.6	Max Throughput (MB/s)	Per Packet (usec)	MPSE (usec)
AC	480.7	9.71	3.69
Intel	510.3	8.94	2.73
% Gain	6.25%	7.92%	26.0%

## Live Traffic Test: Throughput and Latency

Of note, there was more than a 25 percent improvement in the MPSE time within Snort, which resulted in a 6.25 percent increase in the maximum throughput of the system and an 8 percent reduction in the per packet processing time (latency).

In addition, further performance gains can be expected if the software is run on a hardware platform that supports Intel QuickAssist Technology.

## Summary and Future

The quick integration effort described in this paper was a simple replacement of existing Snort rule functionality; however, much more can be done to allow Snort to take advantage of the Intel QuickAssist Technology library. In particular, both the cryptography and data compression elements would increase Snort performance and attack coverage.

Snort could leverage the SSL decryption capabilities to inspect encrypted web traffic for attacks targeted both at web servers and web browsers. Similarly, the performance of the Snort HTTP Gzip/Zip decompression area can be increased by using the compression API, which frees up CPU resources for processing additional packets and reducing latency.

Additionally, there are other areas where the Pattern Matching API would prove useful. The Pattern Matching API supports evaluation of regular expressions, allowing Snort to offload PCRE expression evaluation.

## For More Information

For more information on Intel in Embedded and Communications, please visit <http://www.intel.com/embedded>.



Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Intel® products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications. Intel may make changes to specifications and product descriptions at any time, without notice.

Information regarding third party products is provided solely for educational purposes. Intel is not responsible for the performance or support of third party products and does not make any representations or warranties whatsoever regarding quality, reliability, functionality, or compatibility of these devices or products.

Copyright © 2010 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

Copyright © 2010 Sourcefire, Inc. All rights reserved. Sourcefire, the Sourcefire logo, and Snort are registered trademarks of Sourcefire, Inc.

\*Other names and brands may be claimed as the property of others.