SOURCEfire network SECURITY

SNORT™ 2.0

Hi-performance Multi-rule Inspection Engine

Sourcefire, Inc.
9212 Berger Road
Suite 200
Columbia, MD 21046

April 2004

www.sourcefire.com

TABLE OF CONTENTS

Authors: **Marc Norton**, Sourcefire, Inc.
Daniel Roelker, Sourcefire, Inc.

ABSTRACT

Snort 2.0 introduces a new High Performance Multi-Rule Inspection Engine responsible for detecting rule matches during packet processing. Packets are first analyzed by the Rule Optimizer to select the appropriate set of rules for inspection. Then the Multi-Rule Inspection Engine searches for rule matches, builds a queue of detected rule matches, and selects the best rule match to log based on a simple set of event priorities.

The process of inspecting network traffic for rule matches is performed by using three stages:

1. Rule optimization to produce efficient rule sets for inspection.
2. Set based inspection algorithms that perform high-speed multi-pattern content searches.
3. Parameterized inspection techniques which allow for complicated parameter inspections.

The rule optimization and multi-pattern inspection provides the high performance needed for modern high speed networks. The use of Snorts' standard parameterized inspection allows the rule language to be enhanced without affecting the high performance inspection engine. The combination of these strategies applies the strength of each strategy where it works best, and allows the Snort rule language to remain flexible for future enhancements.

Snort's Multi-Rule Inspection Engine is capable of inspecting Gigabit speed networks without packet loss, while detecting and logging events using very large rule sets.

PARAMETERIZED RULE INSPECTION

A parameterized search sequentially tests each parameter. If any parameter is not accepted as valid the search has failed. Snort originally matched rules against network traffic by performing a parameterized search where the parameters of the search are the Snort rule options. The complexity and performance of such a search is generally a function of the number rules, the number of parameters of each rule, and the size of the combined address space of all of the parameters.

Typically, a parameterized search will use a parameter search order based on each parameter's relative ability to terminate a useless search quickly or narrow the search space efficiently. The actual parameter ordering is usually based on a deeper understanding of the problem being parameterized. There is often one or more parameter orderings that clearly provide a near optimal search.

Snort rules are grouped using several primary parameters such as IP addresses and protocol ports. After a packet is shown to match the primary parameters the rules in that group are than tested one at a time. Snort originally used a parameterized search to walk each parameter in each rule and evaluate whether a given packet or stream matched all of that rules parameters. The strength of a parameterized inspection is that it allows you to have a flexible inspection language. The cost of parameterized inspection is linearly proportional with respect to the number of rules. This presents a performance problem for a large number of rules.

SET BASED RULE INSPECTION

A set based search differs from a parameterized search as follows; a parameterized search processes one rule at a time and tests each parameter of the current rule being tested, a set based search processes one or more parameters at a time and tests all rules in parallel. This allows an efficient set based search to potentially be much less costly than a parameterized search.

Snort uses the content and uricontent fields of the rules to perform a one parameter set based search using a multi-pattern search algorithm. These parameters are very efficient at eliminating packets and streams from further consideration. They are particularly amenable to high performance parallel inspection.

COMBINING SET BASED AND PARAMETERIZED RULE INSPECTION

A set based search can be a highly efficient search technique. Currently Snort uses an initial set based inspection that uses only one parameter, the rule content or uricontent. Once the set based inspection has identified a rule match we must test the rest of the rules parameters against the stream or packet. Since there is only one rule to test against we perform a standard sequential test of each parameter, as originally done by Snort.

RULE SETS & SEARCH TYPES

There are four categories of rules which may need to be tested against a packet. The four categories are further divided into two search types; those with content, and those without content. There are protocol field rules, generic content rules, packet anomaly rules, and IP rules. These are described below:

1. **Protocol Field Rules**
   The protocol field search allows a signature to specify a particular field within a protocol to search for a content match. For example, Snort uses the 'uricontent' keyword to search HTTP request URI fields.
2. **Generic Content Rules**
   The generic content rules allow a signature to specify a generic search of all packet data for a string or byte sequence to match against. For example, this functionality may be used to look for any ASCII or binary strings that may trigger an event, such as 'porn star', or 'root', or 'passwd'.
3. **Packet Anomaly Rules**
   The packet anomaly rules specify characteristics of a packet or packet header that is cause for alarm. These rules have no content. Packet anomaly rules do not require any type of content search, instead they are focused on a packet's other characteristics. While all rule types can include aspects of anomaly detection, the packet anomaly search is specific to rules not requiring any type of content search. As an example, a packet anomaly rule might look for all incoming packets from a specific IP address on a specific port, and may include other constraints on the packet's header fields.
4. **IP Rules**
   The IP rules are applied to the IP protocol layer and may need to be tested against every IP packet. If the packet's protocol is TCP, UDP or ICMP these rules are processed with the corresponding application layer protocol rules. The IP rule search types may be either content or no-content.

CONTENT SEARCHING

Content searching utilizes the High Performance Multi-Pattern Search engine to find all occurrences of each search pattern. Multi-Pattern searching is performed by collecting a group of search patterns and performing a simultaneous search. This type of search can inspect a body of data and find all occurrences of all patterns in the search group. Snort no longer aborts searching on the 1st pattern match. Instead each match is queued, and later processed for event selection after all packet matches are located.

Search patterns correspond to strings defined by the rule 'content' and 'uricontent' keywords. This can cause up to two separate multi-pattern searches to be performed, one for the Protocol Field Rules (HTTP and Telnet) and one for the Generic Content Rules. That is, if both types of rules exist for the specific stream or packet being analyzed.

The Multi-Pattern search is performed as a two stage process. The first stage is the High-Performance Multi-Pattern content search. This finds rule content occurrences in a packet, and identifies which rule the content belongs to, but does not test all of the rules other options against the packet. If the inspection process finds a pattern match, the second stage inspection performs a complete rule inspection using the standard parameterized rule processing. This either validates the match or determines that even though the content of the rule matched a pattern in the packet, the rules' other options failed to match the packet.

During the first stage of the inspection process every search pattern contained in the selected rule set is scanned for in parallel, using a set based multi-pattern search engine. The Multi-Pattern Search Engine currently includes two different set based Multi-Pattern search algorithms. A Modified Wu-Manber [1] style search algorithm and an Aho-Corasick [2] style search algorithm. There is also a naïve set based Boyer-Moore inspection engine for processing rule sets with less than 10 rules.

Multi-Pattern Search Algorithms

The Wu-Manber search algorithm is the default, and uses the least amount of memory. It uses hashing to create sub-groups of rules, and a set based Bad Word Shift to accelerate the search by providing a byte skipping feature. It is a very flexible algorithm and has been used in various UNIX utilities over the years. Due to the Bad Word Shifting feature, rule sets with a larger minimum pattern size can significantly improve this algorithms performance. Each rule group requires about 128K of overhead memory to manage the rule group.

The Aho-Corasick search algorithm is provided as an alternate inspection engine. It uses 2-3x the memory of the Wu-Manber search algorithm. This is a classic implementation of the Aho-Corasick algorithm and does not include a Boyer-Moore shift to skip characters. This algorithm is less sensitive than the Wu-Manber algorithm to small or large minimum pattern size. Though, a larger minimum pattern size can improve the algorithm's performance. Memory consumption is directly governed by the number of bytes used in the patterns. This algorithm has been used and adapted for database and keyword searches where a very large number of keywords are used.

Generally, the Wu-Manber algorithm is the preferred choice for Snort. The Aho-Corasick algorithm can be slightly faster than the Wu-Manber algorithm, but does consume more memory. When considering performance of Multi-Pattern search algorithms several issues must be considered.

1. Intrinsic Algorithmic Performance Characteristics - Performance is rule driven or data driven
2. CPU Architecture Performance – Memory access costs, cache miss costs, relative shift/add/compare/jump operation costs.
3. Compiler and Optimizations Used
4. Application Memory Requirements and Platform Limits

The naïve set based Boyer-Moore algorithm is used for small rule sets. Testing has shown that for 10 rules or less this method usually performs as well or better than the other Multi-Pattern search algorithms.

Validating Rule Matches

The Multi-Pattern search algorithm provides an initial rule match based on the content or uricontent parameter. At this point the content and the unique parameters processed by the Rule Optimizer are matched. The rest of the rule is then tested against the packet to determine if a complete rule match has been found. This testing is done using Snort's full parameter processing.

NO CONTENT SEARCHING

The no-content packet search uses the no-content rules. The no-content rules are processed using a the standard Snort parameterized search scheme. The no-content rules test the network packet header fields for anomalous values. While this search utilizes the parameterized search strategy directly, it remains a very fast search.

EVENT QUEUE PROCESSING

The event queue is used to store information about rule matches as they occur during the multi-rule search. When each packet has been fully inspected for rule matches the events in the Event Queue are processed and a single event is selected for logging.

Queuing Events

Each rule match is considered an event. Since there may be several events associated with an individual packet, each event is queued until the entire packet is fully processed. The events in the queue may correspond to any of the Snort rules. The event queue is flushed after each packet is processed and a final event is selected for logging.

Event Selection

Currently Snort logs a single event for each packet. Therefore a single event must be selected from the event queue. The selection process is based on the following priorities:

Event processing order; log, alert, or pass

Content events supercede anomalous events

Protocol content events such as HTTP events, supercede generic content events

Longer content length matches, if present, supercede shorter content length matches.

ANTI-DOS STRATEGIES

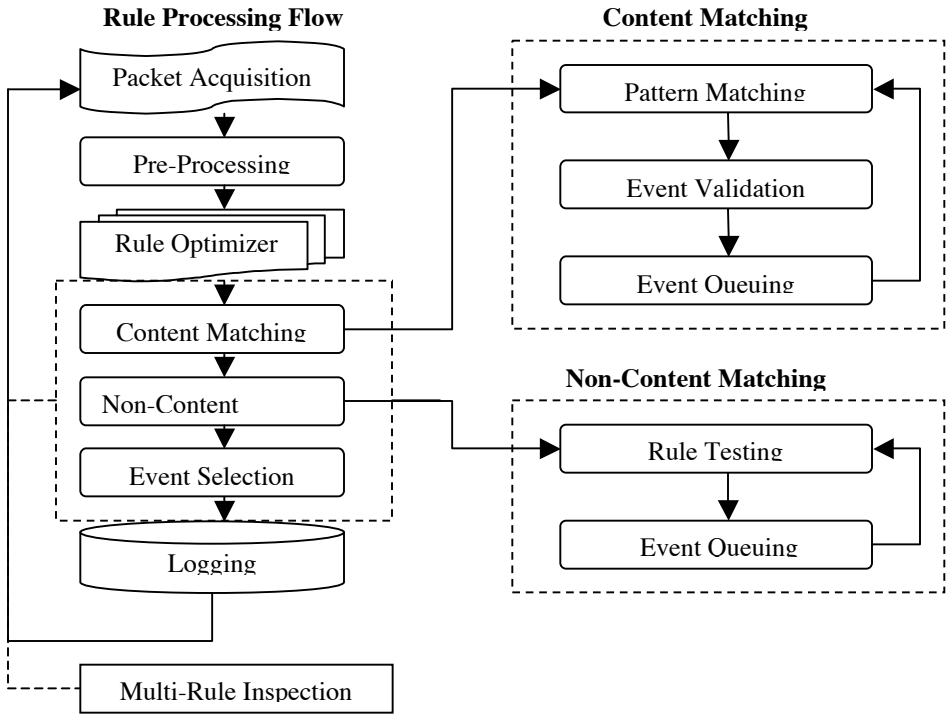
Multi-Pattern search algorithms find all occurrences of all patterns within each searched packet. This presents a natural opportunity for Denial of Service attacks. If the search engine triggers on all occurrences, than any repeated pattern that corresponds to an attack pattern can be used to DOS a naive multi-pattern matcher. It could also be a problem for any pattern matching engine that catches all rule matches.

Therefore, an anti-DOS strategy was built into the high performance inspection process. The strategy goes as follows:

1. Flag each rule tested during packet inspection
2. Do not test flagged rules against the packet any more.
    a. Either the rule tested positive - and we already queued the event, or
    b. The rule failed to match - and there is no further need to test it.

Packets are processed sequentially looking for rule content or uricontent pattern matches, when a match is encountered it is flagged and validated, and is essentially no longer used with the current packet search.

RULE AND PACKET PROCESSING FLOW DIAGRAM



BIBLIOGRAPHY

1. A FAST ALGORITHM FOR MULTI-PATTERN SEARCHING
   Sun Wu, Udi Manber, May 1994
2. Efficient String Matching: An Aid to Bibliographic Search
   Alfred V. Aho and Margaret J. Corasick, Bell Laboratories
   Communications of ACM 18, June 1975, pp333-340
3. Algorithms On Strings, Trees, and Sequences
   Dan Gusfield, Cambridge University Press, 1997
4. A fast string searching algorithm.
   R.S. Boyer and J.S. Moore
   Communications of ACM, 20(10):762-772, 1977