PAPER Special Section on New Technologies and their Applications of the Internet

Reducing Payload Inspection Cost Using Rule Classification for Fast Attack Signature Matching

Sunghyun KIM[†], *Nonmember and* **Heejo LEE**[†a)], *Member*

SUMMARY Network intrusion detection systems rely on a signature-based detection engine. When under attack or during heavy traffic, the detection engines need to make a fast decision whether a packet or a sequence of packets is normal or malicious. However, if packets have a heavy payload or the system has a great deal of attack patterns, the high cost of payload inspection severely diminishes detection performance. Therefore, it would be better to avoid unnecessary payload scans by checking the protocol fields in the packet header, before executing their heavy operations of payload inspection. When payload inspection is necessary, it is better to compare a minimum number of attack patterns. In this paper, we propose new methods to classify attack signatures and make pre-computed multi-pattern groups. Based on IDS rule analysis, we grouped the signatures of attack rules by a multi-dimensional classification method adapted to a simplified address flow. The proposed methods reduce unnecessary payload scans and make light pattern groups to be checked. While performance improvements are dependent on a given networking environment, the experimental results with the DARPA data set and university traffic show that the proposed methods outperform the most recent Snort by up to 33%.

key words: *intrusion detection system, signature matching, rule classification, pattern matching*

1. Introduction

Intrusion detection is a set of techniques and methods used to detect suspicious activities, both at the network and host level. The intrusion detection process aims to find data packets that contain known intrusion-related signatures or anomalies related to the Internet protocols. Intrusion detection methods fall into basic categories: signature-based detection and anomaly-based detection.

Signature-based detection is used to compare against activity in the network or host with predefined signatures produced by an analysis of an attack or malicious packets. This method relies on a database of attack signatures. Therefore, it is only as effective as its database. Most signatures have patterns to search known attacks. Anomaly-based intrusion detection, by contrast, utilizes a more generalized approach when searching for and detecting threats in a network. A rule of normal behavior is developed and when an event falls outside that norm, it is detected and logged. The behavior is a characterization of the state of the protected system; a reflection of the system health, and is sensitive to attacks. In this context, an anomaly-based method of intrusion detection has the potential to detect new or unknown attacks.

Intrusion detection methods are used in Network Intrusion Detection System (NIDS) [1], [2], that captures data from the network and applies rules to the data for detecting anomalies. NIDS detects malicious activities, such as denial of service attacks, port scans or even attempts to hack into computers by monitoring network traffic. Based on a set of signatures and rules, after a match is found, NIDS takes specified actions, such as logging the event or sending an alarm to a management console. Studies on NIDS have attempted to rapidly decide which packets are malicious. A large number of signatures are associated with well-known ports and there is an increase of network data volume; the cost of scanning payload is increasing. It is reasonable to check the protocol fields before comparing patterns in the payload to avoid unnecessary pattern matching. Thus, we proposed new methods, whereby the protocol fields have a higher priority than the packet payload for signature matching. Proposed methods are similar to research on rule clustering by protocol fields such as a decision tree [3] or an evaluation tree [4]. A large rule set, since they use tree structure, need much memory to maintain data nodes. Thus, they are inadequate for a large rule set. However, the proposed methods are designed to deal with a large rule set. Despite multi-dimensional classification, they require less memory to classify rules. They make light pattern and perform payload scan only when packet's protocol fields used in classification are matched. The contribution of this study is as follows:

- From analysis of Snort' rules, we found that the number of rules are increasing every year and many rules are concentrated at specific ports. To cope with many rules, we show the necessity of rule classification that can reduce the number of patterns to be matched.
- We propose new methods to classify rules with multiple protocol fields for reducing pattern matching cost. They support multi-dimensional classification and easily tune the degree of classification. In our experiments, when all Snort's rules are used, our methods can reduce memory usage for pattern matching from 30% to 43% and shorten detection time for rules from 2% to 33% compared to unmodified Snort.
- We analyze signature detection costs and show their change based on various dimensional classifications of rules. The rules classification by network flow per-

Manuscript received February 3, 2009.
Manuscript revised May 20, 2009.

[†]The authors are with Korea University, Seoul 136–713, South Korea.
a) E-mail: heejo@korea.ac.kr

DOI: 10.1587/transinf.E92.D.1971

forms well under general IDSes deployment and current rule distribution.

The remainder of this paper is organized as follows. Section 2 briefly outlines related work. Section 3 explains the necessity of rule classification. Section 4 describes the proposed methods. Section 5 analyzes the performance and experimental results. In Sect. 6, we summarize our experience.

2. Related Work

Just as a network packet consists of the header and the payload, the research on signature matching can be classified into two. First, a pattern matching for a packet data that mainly consists of string matching. Second, the classification of a rule set by the protocol fields. The latter focuses on reducing the number of rules to be searched by grouping, that is, classification or clustering. The former focuses on the means to rapidly search certain strings. We will briefly discuss some of the methods for signature matching and explore Snort's internals.

Several pattern matching algorithms have been proposed for the payload matching. The Boyer and Moore algorithm [5] is a well-known the single pattern matching algorithm. It preprocesses the target string that is being sought, to generate a table of mismatch skip values based on the pattern position involved in the mismatch. Another well-known algorithm, The Knuth-Morris-Pratt (KMP) [6] also preprocesses patterns, to generate a look-up table that indicates how many positions the pattern can be shifted to the right based on the position in the pattern where a mismatch occurs. The multi-pattern matching method searches a text string for the occurrence of any pattern in a set of patterns, using only a single iteration. The well-known Aho-Corasick (AC) algorithm [7] preprocesses the set of patterns, to construct a pattern matching machine based on a deterministic finite automaton (DFA). The Wu-Manber algorithm [8] is based on the bad character heuristic, which is similar to Boyer-Moore, but uses a one or two-byte bad shift table constructed by re-processing all the patterns, instead of only one. Another method, Exclusion-based signature Matching (E^2xB) [9] is designed to provide rapid negatives, when the search string does not exist in the packet payload. Fisk [10] and Antonatos [18] compared several multi-pattern matching algorithms.

Other researches on rule classification have progressed. Kruegel and Toth proposed a decision tree method to improve signature-based intrusion detection [3]. This method uses a well-known clustering algorithm that is applied in machine-learning to create an optimized decision tree, which is used to find malicious events, using a minimum of redundant comparisons. It builds a decision tree from a classified set of data items with different features using the notion of information gain. Sinha et al [4] proposed an evaluation tree that determines which rule groups are maintained in memory by choosing protocol fields and values recursively. Initially, the method selects the protocol field that is most effective in rejecting the rules, and then separates those groups by values of the chosen protocol field. After forming groups for each of these values, the algorithm recursively splits the groups by other protocol fields that reject at least a threshold number of rules, producing smaller groups. By this means, it generates a hierarchy of protocol fields and values using a tree structure. These methods encounter difficulties if there is a large number of rules. As the number of rules increase, many leaf nodes are created. This implies that the same number of pattern groups are made to use multi-pattern matching. Each pattern group needs rule replication to avoid multiple payload scan. As rules increase, leaf nodes increase and rule replication also increases. If they have a large number of rules, these methods require a large mount of memory for pattern groups. Thus, they cannot properly handle a large number of rules due to memory shortage.

One of the NIDSes, Snort [1], is an open source network intrusion prevention and detection system utilizing a rule-driven language that combines the benefits of protocol and payload signature. Snort is commonly used to actively block or passively detect a variety of attacks and probes performing protocol analysis and content matching. Snort considers that rules are composed of two components, a rule header and a rule option. The rule header has predicates of the protocol fields. The rule option has mainly strings for pattern matching and other predicates. After parsing the rules, based on the port, Snort makes three types of pattern groups that consist of destination port groups for rules having a unique destination port numbers, source port groups for rules having a unique source port number and a generic port group for rules without a unique destination port number and source port number. A generic port group is copied to other port groups for efficiency. Snort provides the Wu-Manber and the Aho-Corasick pattern matching algorithm. When packets are going through, based on port number, multi-pattern matchers of the corresponding port group are called. Based on the source port and destination port, packets are scanned once or twice for the destination port group or the source port group, or both. Snort only uses one protocol field, source port or destination port, for grouping rules. Under a heavy payload or a large number of patterns, a great deal of time is required to inspect the payload. The proposed methods are devised to reduce the cost of payload scanning classifying patterns with multiple protocol fields.

3. Reducing Payload Scan Cost

New attacks are created all the time, and as they are, new rules are added to the IDSes' rules. The cost for signature matching consists of the cost of pattern matching and the cost of checking protocol fields' predicates. The pattern matching problem in IDSes requires consideration of many issues associated with pattern searches, for example, multi-pattern search algorithms, pattern sizes, pattern group size, alphabet size, search text size, and frequency

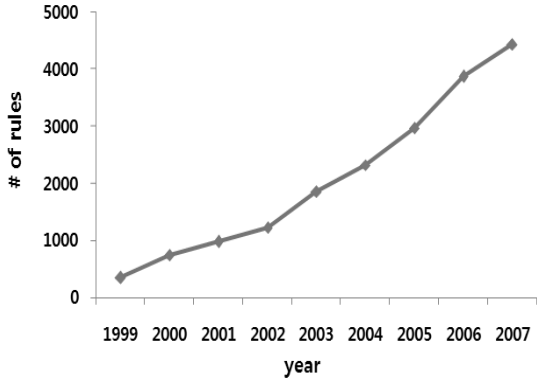


Fig. 1 The growth of Snort rules having CVE identifiers.

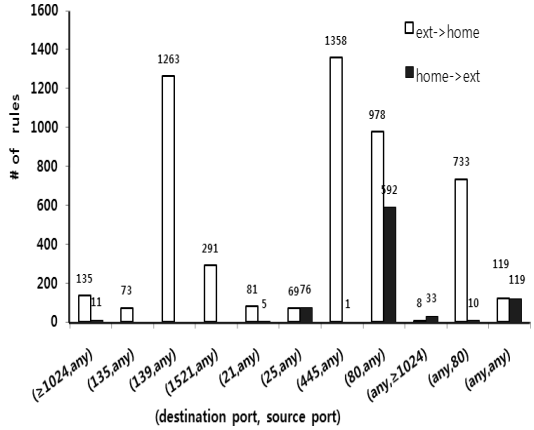


Fig. 2 The classification of Snort rules related to traffic flow and ports.

of searches. Some are external factors that are irrelevant to the detection engine, such as alphabet size, search text size, and pattern size. Others are internal factors that have a close relation to the detection engine, such as multi-pattern search algorithms, pattern group size, and frequency of searches. Among external factors, search text size and frequency of searches have been increasing because of the growth of various applications generating much network traffic. Formenkov and McCreary observed [11], [12] that the average packet length was over 400 bytes and 40% of packets were over 400 bytes, at that time, in spite of fewer applications existing than is the case of now. The recent explosive growth of Internet users and data increases both packet payload and network traffic.

We analyzed snort's rules [1] (VRT Certified Rules for Snort v2.7) to determine the change of pattern group size in attack signature. Figure 1 shows the size of the Snort rules has grown over time. To make matters worse, a large number of signatures are associated with specific ports. 4935 (70%) rules are associated with three destination ports (80,445,139) and one source port (80). The skewness of rule distribution can be used in algorithmic attacks. Also according to H. Dreger et al [13]'s protocol analysis, some applications do not use standard ports. Therefore, IDSes have to check many patterns if rules are simply classified by destination port or source port. That is, the necessity of rule classification is as follows:

- Attack signatures are rapidly increasing with the increase in threats (e.q. malicious code, application's exploits, and hacking). Many attacks use a wide range ports rather than specific port. Without proper rule classification, they create big pattern groups.
- The explosive increase of Internet usage and growth of applications generating heavy traffic, IDSes have not only to search long text sizes but to increase the frequency of searches.

If patterns are grouped by multiple fields, not only can we make small-sized pattern groups, but we also reduce the search frequency. Multi-dimensional classification requires a large amount of memory to build many multi-pattern matchers due to numerous rule duplications. Also it requires overhead to choose classified pattern groups. IDSes are generally deployed between a gateway router and a backbone switch or between firewall and backbone switch. Therefore, IDSes can simplify every packet flow into two types. Among the 6985 default rules related to TCP in Snort rules, 5780 (84%) rules are applied when traffic comes from external networks to the home network and only 1103 (16%) rules are applied when traffic goes out from the home network to external networks (Fig. 2). So if we classify patterns by rules adapted for outgoing or incoming packet, we can reduce the number of patterns to be inspected and the probability of the payload scan. Based on this analysis of rules, we propose new rule classification methods.

4. Multi-Dimensional Rule Classification

IDSes' rules have increased rapidly over recent years. Simple classification method as Snort does requires a heavy multi-pattern matching engine. Adding protocol fields for rule classification, the proposed methods build a light multi-pattern matcher and avoid unnecessary payload scans. We shall explain these methods in more detail.

4.1 Address Group Classification and Detection

AGC (Address Group Classification) uses two dimension classification of the two protocol fields. These fields are the destination port and source address or the source port and source address. Let R be the set of n rules, i.e., $R = \{r_1, r_2, \ldots r_n\}$. Let $F = \{f_1, f_2, \ldots, f_m\}$ denote the set of m protocol fields present in the rule set. Let $P_{f_i} = \{p^1_{f_i}, p^2_{f_i}, \ldots, p^k_{f_i}\}$ denote the set of the predicates associated with f_i in the rule set. Also let $V_{f_i} = \{v^1_{f_i}, v^2_{f_i}, \ldots, v^j_{f_i}\}$ denote the set of unique values extracted from f_i's predicates used in R in ascending order. If we let $f_{dp}, f_{sp}, f_{sip}, f_{dip}$ denote destination port, source port, source IP address and destination IP address, AGC classifies rules with $v^i_{f_{dp}}$, $v^j_{f_{sip}}$ or $v^i_{f_{sp}}$, $v^j_{f_{sip}}$.

As observed earlier, due to their deployment, IDSes

can see only two network flows. Thus, AGC divides IP addresses used in rules into two address groups, "HomeExt" group and "ExtHome" group to reduce the complexity and the multiplicity of classification. Most of the IDSes rules can be classified with such address groups. "HomeExt" group has the source address belonging to home network and the destination address not belonging to home network, in other words, belonging to external network. "ExtHome" group has source address and destination address opposite to the "HomeExt" group. Rules having specific network addresses or host addresses are also divided two groups, as "HomeExt" group or "ExtHome" group includes such addresses. So, AGC merges specific addresses into two groups and creates two pattern groups in each destination port or source port. After pattern matching, if any pattern is matched in the packet payload, AGC verifies the specific addresses by checking if the packet's address is matched with rule's address predicates. Since AGC assumes that the destination address is opposite to the source address, it checks only the source address or the destination address and decides the appropriate pattern groups based on packets' port number. Like Fig. 3, In the rule parsing stage, AGC classifies patterns into each port groups based on rules' port numbers. The rules without a specific port number are copied both port groups and generic port group. Then, AGC divides them into two address groups, "HomeExt" and "ExtHome", according to source IP address. After rules grouping, the multi-pattern matchers are built with the classified patterns. When a packet passes through the signature detection engine, the appropriate multi-pattern matchers are chosen based on its port numbers and source address. Depending on packet's port numbers and source address, the payload scan is performed once or twice which means rules can be adapted once or twice. For example, if the packet's port numbers and source address are matched both of two patter groups classified with ports numbers and source IP addresses, payload scan happens twice. But if the packet's port numbers and source address are matched with single port groups or is not matched both of them, payload scan happens once. AGC is similar to Snort's way except making the additional classification with network flow. AGC reduces the chance of payload scan and makes light pattern matching engine compared to Snort's way.

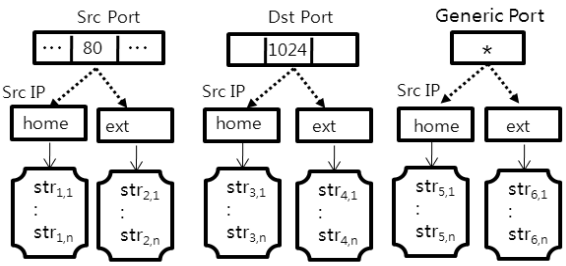


Fig. 3 Example of AGC.

4.2 Unified Predicates Classification and Detection

UPC (Unified Predicates Classification) provides various multi-dimensional classification. It creates a fully classified pattern matcher and scans packet's payload only once. A rule in R can be described as the relationship of predicates of protocol fields like $r_t = p^i_{f_1} \wedge p^j_{f_2} \wedge \ldots . p^k_{f_m}$. Therefore, a predicate used in R can be presented like $p^j_{f_i} = f_i \otimes v^k_{f_i}$ where $\otimes$ is an operator used in the predicate ($=, \leq, \geq$, etc.), f_i is the i^{th} element of F, and $p^j_{f_i}$ is the j^{th} element of P_{f_i}. Let $s^j_{f_i}$ denote all rules' results depending on an element of V_{f_i}. Let $dom(f_i)$ denote f_i's domain. $dom(f_i)$ can be divided into $(2k+1)$'s sub regions, where $k = |V_{f_i}|$. We describe $S_{f_i} = \{s^1_{f_i}, s^2_{f_i}, \ldots . s^{2k+1}_{f_i}\}$ to present the set of all possible results that depend on all values of f_i. $s^{2j}_{f_i}$ of S_{f_i} has the results of $r_1, r_2, \ldots r_n$ regarding to $v^j_{f_i}$ of V_{f_i}. Likewise, $s^{2j+1}_{f_i}$ of S_{f_i} has the results of $r_1, r_2, \ldots r_n$ depending on all values between $v^j_{f_i}$ and $v^{j+1}_{f_i}$ of V_{f_i}. Since we know $v^j_{f_i}$ from rules, we can calculate $s^j_{f_i}$. Also we can compute $s^{2j+1}_{f_i}$ with any value between $v^j_{f_i}$ and $v^{j+1}_{f_i}$ in advance. Therefore we can find all the results of the rule set by each element of f_i used in the rule set. Based on values of f_i, P_{f_i} is decided and depending on P_{f_i}, matched rules among $r_1, r_2, \ldots r_n$ are determined. The number of rules' results including f_i' predicates is $2 * n|V_{f_i}| + 1$. We present S_{f_i} in bitmap and store them in an array data structure called PFA (Protocol Filter Array). UPC uses several PFAs and seeks all bitmap combinations.

PFA for f_i provides the integrated processing of predicates and results of the rule set related to f_i. UPC finds all combinations of bitmaps in PFA for $f_1, f_2, \ldots f_m$. Then UPC performs the conjunctive operation (&) with each bitmap to find all possible results of the rules. In the worst case scenario, the maximum number of these results is 2^n, if there are n rules. Based on the final bitmap, UPC decides which rules' pattern should be grouped together. For example, we have two rules similar to Snort's rule, such as the following.

r_1: alert tcp \$HOME_NET 25 –>\$EXTERNAL_NET 80 . . . ;content:login;. . .

r_2: alert tcp \$EXTERNAL_NET 21:25 –>\$HOME_NET 143 . . . ;content:root;. . .

Figure 4 details the 3-dimensional classification process using three PFAs that are protocol fields used in r_1 and r_2. If a packet's destination port is between 21 and 24 only

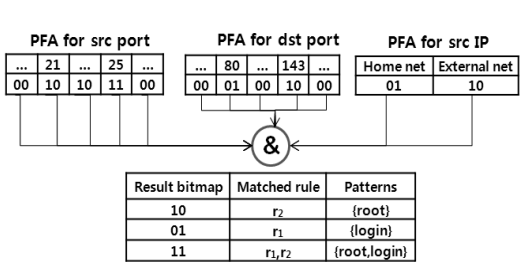


Fig. 4 Example of UPC.

r_2 is matched. For a packet's destination port 25, both rules are matched. In other cases, both rules are not matched. PFA has the rules' result as bitmaps. After the combination of three PFAs' bitmap, UPC obtains three resulting bitmaps, such as '01', '10', '11'. Based on the resulting bitmaps, UPC makes three small-sized pattern groups. Also UPC can easily tune the degree of rule classification by the number of PFAs. When many PFAs are used, UPC makes light but many patten groups and needs much memory because of rule duplications. So UPC merges pattern groups with a small number of patterns, and makes appropriately sized pattern groups to reduce memory.

As has been demonstrated, UPC pre-calculates all the rules' results and saves them into PFAs. When a packet reaches the detection engine, based on the value of the packet's protocol field, the detection engine searches the results of the rules in PFAs. After obtaining all corresponding result bitmaps of the PFAs, the final result is decided by & (AND) bit operation to each bitmap. Based on bitmap value, the detection engine decides if the packet needs payload scans and identifies pattern groups to be adapted. UPC only performs the search operation once, irrespective of the rules' predicates. Therefore, the greater the complexity of rules' predicates, the better the performance. Since it does not have a fixed order, UPC can change the search order of PFAs. For example, if a certain protocol field can drop early normal packets, UPC can change the sequence of PFAs.

5. Evaluation

The proposed methods are based on, in general, pattern matching of payload requiring more processing time than protocol field matching. In the case of TCP, while the packet header has 20 bytes without option, the packet data can have 1460 bytes, considering MTU (Maximum Transmission Unit). The size of the packet payload and values of the protocol fields depend on the network environment. Therefore, if a packet has little or no payload, checking the protocol field first can yield inferior results. If a pattern group has few pattern, since it does not affect performance, the proposed method cannot have any benefits. Conversely, if the packet payload is long and there are many rules, the proposed methods confer benefits. Multi-dimensional rule classification reduces the number of patterns in multi-pattern matchers. Furthermore, it minimizes the packet payload scan, because of checking multiple predicates before performing the payload scan. We analyze the performance of these two aspects.

5.1 Performance Analysis

Rule set, R, has n rules and all rules have each predicate of $f_1, f_2, \ldots, f_m$ and patterns to be matched. As noted above, V_{f_i} is the set of unique values extracted from a protocol field f_i's predicates used in R. In the case of a single protocol field classification, the average pattern count of groups is $\frac{n}{|V_{f_i}|}$, However, if we make pattern groups with k protocol fields, such as $f_1, f_2, \ldots f_k$, the average pattern count of groups is $\frac{n}{\prod_{i=1}^{k}|V_{f_k}|}$. Decrease of the average number of patterns to be matched reduces pattern matching cost. Also, the probability of payload scan is lowered in the same manner.

We denote $P(f_i)$ the probability for a packet to pass f_i's predicates. Let C_{f_i} denote the cost for a packet to decide f_i's predicates. $C_{mp_{(gen,f_1,\ldots f_m)}}$ is the scan cost of a multi pattern group that classified by $f_i, \ldots f_m$ with generic rules. $C_{mp_{(f_1,\ldots f_m)}}$ is the scan cost of a multi pattern group that classified $f_i, \ldots f_m$ without generic rules. $P(mp)$ is the probability of finding patterns in the a packet's payload. The signature detection cost using multi-pattern matching can be divided in three parts. The first is the cost of checking predicates before pattern matching, which is used to identify proper pattern groups. The next is the cost of pattern matching itself. The last is the cost of verifying predicates after pattern matching. In the same way, we can get each method's cost of signature matching as follows:

$$C_{Snort} = \sum_{i=1}^{2}(C_{f_i} + P(f_i)\cdot C_{mp_{(gen,f_i)}}) + \prod_{i=1}^{2}(1-P(f_i))\cdot C_{mp_{(gen)}} + P(mp)\cdot\sum_{i=1}^{m}(C_{f_i}\cdot P(f_{k-1})) \quad (1)$$

$$C_{AGC} = \sum_{i=1}^{2}(C_{f_i} + C_{f_3} + P(f_i)\cdot P(f_3)\cdot C_{mp_{(gen,f_i,f_3)}}) + \prod_{i=1}^{2}(1-P(f_i))\cdot C_{mp_{(gen,f_3)}} + P(mp)\cdot\sum_{i=1}^{m}(C_{f_i}\cdot P(f_{i-1})) \quad (2)$$

$$C_{UPC} = \sum_{i=1}^{k}(C_{f_i}\cdot P(f_{i-1})) + \prod_{i=1}^{k}P(f_{i-1})\cdot C_{mp_{(gen,f_1,\ldots f_k)}} + P(mp)\sum_{j=k}^{m}(C_{f_i}P(f_{j-1})) \quad (3)$$

where $P(f_0) = 1$, f_1 is the destination port, f_2 is the source port, and f_3 is the source IP address. Whereas k dimensional classification needs the sum of overhead for k protocol fields checking before pattern matching, the chance of pattern matching is decreased by the product of passing probability of k protocol fields' predicates. In addition, multi pattern matching cost C_{mp} decreased in proportion to the combination of k protocol fields, as discussed above.

For simplicity, we assume that $P(mp)$ is too small, so that we ignore the last part among three parts of the signature detection cost. We can simplify Eq. (1), Eq. (2), and Eq. (3) as follows:

$$\begin{aligned} &if\ \forall P(f_i) = 0\ and\ \forall C_{f_i} = \sigma,\ (1 \le i \le k)\\ &C_{Snort} = 2\cdot\sigma + C_{mp_{(gen)}}\\ &C_{AGC} = 4\cdot\sigma + C_{mp_{(gen,f_3)}}\\ &C_{UPC} = k\cdot\sigma + C_{mp_{(gen,f_1,\ldots,f_k)}} \end{aligned} \quad (4)$$

$$
\begin{aligned}
&\text{if } \forall P(f_i) = 1 \text{ and } \forall C_{f_i} = \sigma,\ (1 \le i \le k)\\
&C_{Snort} = 2 \cdot \sigma + C_{mp_{(gen,f_1)}} + C_{mp_{(gen,f_2)}}\\
&C_{AGC} = 4 \cdot \sigma + C_{mp_{(gen,f_1,f_3)}} + C_{mp_{(gen,f_2,f_3)}}\\
&C_{UPC} = k \cdot \sigma + C_{mp_{(gen,f_1,\ldots,f_k)}}
\end{aligned} \tag{5}
$$

Equation (4) is the case of one payload scan and Eq. (5) is the case of two payload scan. From Eq. (4) and Eq. (5), we can easily find the overhead and the benefits of each method. AGC and UPC has more overhead than Snort, but based on pattern matching cost, they could be better or worse. In the best case for UPC like Eq. (5), UPC needs one payload scan for patten matching but Snort and AGC requires two payload scan for pattern matching. Also while the UPC' pattern group has one copy of the generic rules' patterns, Snort and AGC's pattern groups have two copies of generic rules' patterns.

Based on the previous rules analysis, AGC classifies "HomeExt" and "ExtHome" address groups that have 16% and 84% of the rules, respectively. Let t_{in} denote the percentage of incoming traffic. In the case of AGC, the number of patterns to be matched in each source port or destination port is decreased by $(t_{in} * 0.84 + (1 - t_{in}) * 0.16)$. If there is much outgoing traffic, performance is much better than the opposite traffic flow. The performance of the proposed methods is dependent on the network environment and the distribution of rules.

5.2 Experiments

We have modified Snort version 2.7.0.1 to implement the proposed methods. The experimental platform is a personal computer with an Intel Core2 2.13 Ghz CPU and 2 Gbytes RAM. We used the Linux operating system, Fedora 6. The rule set used for the experiments consisted of only the TCP rules among the VRT Certified Rules for Snort version 2.7. As Allen points out in [14] and Mell states in [15], successful development and testing of IDSes' performance requires robust and accurate training data.

We used two kinds of data files. One is well-known data sets, the DARPA Intrusion Detection Evaluation Data Set from MIT Lincoln Lab [16]. The other is the university traffic gathered at three campuses, which are represented "univ1", "univ2" and "univ3" respectively. Table 1 summarizes characteristics of data files used in our experiments. "nt" and "ddos" are inside traffic among the 2000 DARPA Data Sets and "week1" and "week2" are one-day traffic from inside traffic of 1999 DARPA Data Sets. "% (in:out)" means percentage of packets going from the home network to the external network vs. percentage of packets coming from external network to the home network. "% (= 80)" is the percentage of packets with destination port number equal HTTP (80). "% (≥ 1024)" is the percentage of packets with destination port number equal or greater than 1024. We removed internal home network traffic from all data files, because we assumed that IDS is deployed between a gateway router and a backbone switch. After the proposed methods were executed multiple times, we recorded the average detection time. In experiments, we created four PFAs for destination port, source port, destination IP, and source IP.

Table 2 presents memory usage of each method. Adding source address, AGC makes smaller pattern groups than Snort does. When all rules are adapted, UPC creates many pattern groups and needs much memory. Because generic rules are duplicated into every pattern group, they make heavy pattern matchers and also require more memory. When UPC makes multi-dimensional classified pattern matchers, it merges small-sized pattern groups to solve memory issue. Thus, UPC (dp, sp, sip) creates only 98 pattern groups. We sat the threshold for merging patterns to 100. That is, the merged rule groups cannot exceed 100 patterns. We obtained the threshold value from our experiments and Norton's observation [17] which was that merging up to 100 patterns in multi pattern matcher had little influence on pattern matching performance. Our experiments for the threshold were similar to his paper. Figure 5 compared each method when all rules are used in the detection engine. While AGC has good performance under both light and heavy payloads, UPC methods obtain better performance under heavy payloads than light payloads. Since data files do not have traffic from home network to home network UPC (dp, sp, sip, dip) and UPC (dp, sp, sip) have few differences. Among UPC's several combinations

Table 1 The characteristics of university and DARPA data set.

file	size*	payload**	% (≥1024)	% (80)	% (in:out)
week1	512	40	47	22	50:50
week2	371	55	46	23	49:51
nt	211	63	44	21	49:51
ddos	275	78	43	19	51:49
univ1	1,159	308	65	32	48:52
univ2	1,142	700	93	7.8	44:56
univ3	1,048	927	99	0	33:67

* Mbyte, ** byte (average)

Table 2 Pattern groups and memory.

Method	Snort	AGC	UPC (dp, sp)	UPC (dp, sp, sip)
# of groups	359	429	257	98
size (Mb)	879	500	1666	617

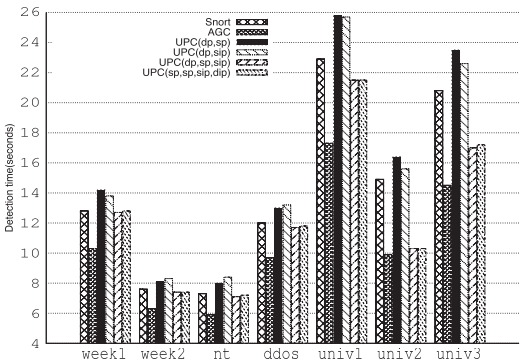
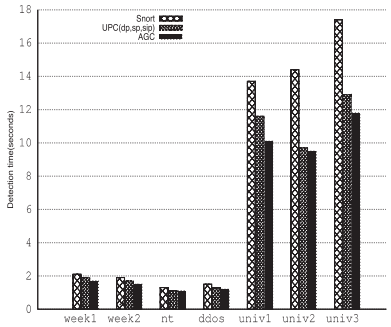
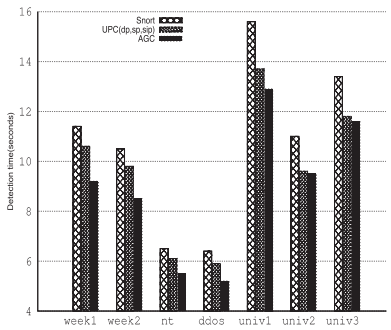


Fig. 5 The detection time of various multi dimensional rule classification.



(a) dst port: ≥1024, src port: any (389 rules)



(b) dst port: 80, src port: any (1,813 rules)

Fig. 6 The detection time of specific rule groups.

of protocol fields, UPC (dp, sp, sip) had good detection time. AGC outperformed UPC (dp, sp, sip) in detection time because most traffic used in experiments need single payload scan. To scan payload once, UPC makes all multiple pattern matchers based on protocol fields in advance. During detection time, UPC needs additional bitmap operations to find the appropriate pattern matcher. If there are short packets having few data, this overhead makes detection performance worse. AGC also requires less memory than other methods because it makes light pattern matchers. We observed the effect on several important port groups that have a large number of rules and frequently appear in data files simultaneously. Figure 6 shows examples of them. When the rule group has a destination port greater than or equal 1024 (389 rules), both AGC and UPC (dp, sp, sip) reduced the processing time to various degrees compared to unmodified Snort. A rule group with destination port HTTP (1813 rules) was very similar to previous port. Both methods have better performance on university traffic that have heavier payload than DARPA traffic, because packet payload size greatly affects the proposed methods' performance.

5.3 Discussion

Because of the increase in exchanged data through networks and attack signatures, IDSes need fast detection engines. Detection engines use multi-pattern matcher for the fast detection of attack patterns. Rule classification is a method to build a light and fast multi-pattern matcher. Points to consider on rule classification are as follows:

- Generic rules form big pattern group due to rule duplication and they need much memory as depth or dimension of classification increases. They reduce the early drop effect from packet filtering.
- The effect of rule classification depends on the protocol fields used in rule classification. We should choose protocol fields which can reduce the probability of pattern matching and also diversify patterns into pattern groups.
- If there are a few rules and short packet payload in the network, the improvement in detection time could be insignificant. In contrast, it may increase only classification overhead. Therefore, we should consider that the number of rules and payload length greatly affects on effect of rule classification.

AGC is a simplified two dimensional classification method that is classified by source or destination port and source IP address. It scans payload once or twice for pattern matching. UPC can do various dimensional classification and scans payload only once regardless of the protocol fields. AGC has two copies of generic rules' patterns (one for source port groups; the other for destination port groups), but UPC's pattern groups have only one copy of generic rules' patterns in every pattern group. In previous experiments, though UPC did not exhibit better performance than AGC, UPC performs better than AGC in the following cases:

- If there are many packets with source port and destination port needed pattern matching or a lot of generic rules, UPC can do faster pattern matching than AGC.
- While AGC simply classifies incoming or outgoing traffic, UPC can deal with various traffic flows. If security devices are deployed where they can see internal packet flow (from the home network to the home network), UPC can avoid payload scan for such traffic. UPC has lower probability of having to do a payload scan than AGC.

6. Conclusions

We analyzed attack signatures and explained the necessity of rule classification in this paper. Attack signatures are increasing rapidly every year. Thus, pattern matching cost also increases because most attack signatures have patterns to be searched in the packet payload. Over time, we have to inspect more patterns and do so more frequently. Therefore, the effect of rule classification becomes more evident. We proposed rule classification methods that reduce the chance of payload scanning for pattern matching and decrease the number of patterns for packets to be inspected. The proposed methods require less memory and shorten signature detection time compared to a simple classification method, such as Snort. However, the effect of the rule classification

is closely associated with rule distribution and network traffic. It is becoming common for network devices to inspect packet payload. Therefore these methods can be adapted to any applications or devices that need pattern matching such as intrusion detection/prevention systems, firewalls, web proxies, and layer seven switches.

References

[1] M. Roesch, "Snort-lightweight intrusion detection for networks," Proc. USENIX LISA, Nov. 1999.
[2] V. Paxson, "Bro: A system for detecting network intruders in real-time," Comput. Netw., vol.31, pp.2435–2463, Dec. 1999.
[3] C. Kruegel and T. Toth, "Using decision trees to improve signature-based intrusion," Recent Advances in Intrusion Detection, LNCS 2820, pp.173–191, Springer Verlag, 2003.
[4] F.J.S. Sinha and J.M. Patel, "Wind: Workload-aware intrusion detection recent advances in intrusion detection," Recent Advances in Intrusion Detection, LNCS 4219, pp.290–390, Springer Verlag, 2006.
[5] R.S. Boyer and J.S. Moore, "A fast string searching algorithm," Commun. ACM, vol.20, pp.762–772, 1977.
[6] J.H.M.D.E. Knuth and V.R. Pratt, "Fast pattern matching in strings," SIAM J. Comput., vol.6, pp.323–350, 1977.
[7] A. Aho and M. Corasick, "Fast pattern matching: An aid to bibliographic search," Commun. ACM, vol.18, pp.333–340, June 1975.
[8] S. Wu and U. Manber, "A fast algorithm for multi-pattern seaching," Technical Report TR-94-17, Department of Computer Science, University of Arizona, May 1994.
[9] X. Wang and H. Li, "Improvement and implementation of network intrusion detection system," IEEE Trans. Softw. Eng., vol.3, pp.48–52, Jan. 2006.
[10] M. Fisk and G. Varghese, "Fast content-based packet handling for intrusion detection," Tech. Rep., May 2001.
[11] D.M.M. Fomenkov, K. Keys, and K. Claffy, "Longitudinal study of internet traffic in 1998–2003," Winter International Symposium on Information and Communication Technologies, Jan. 2004.
[12] S. McCreary and K. Claffy, "Trends in wide area ip traffic patterns: A view from ames internet exchange," 13th ITC Specialist Seminar, 2000.
[13] M.M.V.P.H. Dreger, A. Feldmann, and R. Sommer, "Dynamic application-layer protocol analysis for network intrusion detection," Proc. 15th USENIX Security Symposium, pp.257–272, July 2006.
[14] W.H. Allen, "Mixing wheat with the chaff: Creating useful test data for ids evaluation," IEEE Security & Privacy, vol.5, pp.65–67, July 2007.
[15] V.H.P. Mell and R. Lippmann, "An overview of issues in testing intrusion detection systems," Technical Report NIST 7007, National Institute of Standards and Technology, Jan. 2003.
[16] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln lab," ACM Trans. Information and Systems Security (TISSEC), vol.3, no.4, pp.262–294, 2000.
[17] M. Norton, "Optimizing pattern matching for intrusion detection," White Paper, Sourcefire, 2004.
[18] M.P.S. Antonatos, K.G. Anagnostakis, and E.P. Markatos, "Performance analysis of content matching intrusion detection systems," Proc. 4th IEEE/IPSJ SAINT, Jan. 2004.

Sunghyun Kim received the B.S. degree in Computer Science from Pukyung University, Korea, in 1994, and the M.S. degree in Computer Science from Yonsei University, Korea, in 2006. Currently, he is a Ph.D. candidate in Computer Science and Engineering, Korea University. His research interest includes network security and security policy.

Heejo Lee is an associate professor at the Division of Computer and Communication Engineering, Korea University, Seoul, Korea. Before joining Korea University, he was at AhnLab, Inc. as a CTO from 2001 to 2003. From 2000 to 2001, he was a postdoctorate at the Department of Computer Sciences and the security center CERIAS, Purdue University. Dr. Lee received his B.S., M.S., Ph.D. degree in Computer Science and Engineering from POSTECH, Pohang, Korea. Dr. Lee serves as an editor of the Journal of Communications and Networks. He has been an advisory member of Korea Information Security Agency and Korea Supreme Prosecutor's Office. With the support of Korean government, he worked on constructing the National CERT in the Philippines (2006) and consultation on cyber security in Uzbekistan (2007) and Vietnam (2009). More information is available at http://ccs.korea.ac.kr