Chapter 3
Pre-processor

3.1 Snort payload rule

3.1.1 Analysis Snort payload conditions

With Snort rules, the *content* payload pattern option, the *pcre* (PERL Compatible Regular Expressions) option and the *uricontent* - URL content option are special payload conditions. Information will be produced by Snort pre-processors when one or more of these conditions are met. The number of times that the keywords *content, pcre* and *uricontent* occur in the different versions of Snort rule sets is illustrated in Table 3-1.

Table 3-1. Appearing count of conditions in several Snort rule sets

Snort version	*content*	*pcre*	*uricontent*	total
2.0	1393	0	569	1962
2.1	1763	616	584	2963
2.2	2228	702	584	3514
2.3 (2005, Apr)	3511	820	585	4916

The keyword *content* is the most general Snort keyword and it appears more frequently with later Snort rule sets. Figure 3-1 shows the growth of Snort rule set over the last five years.

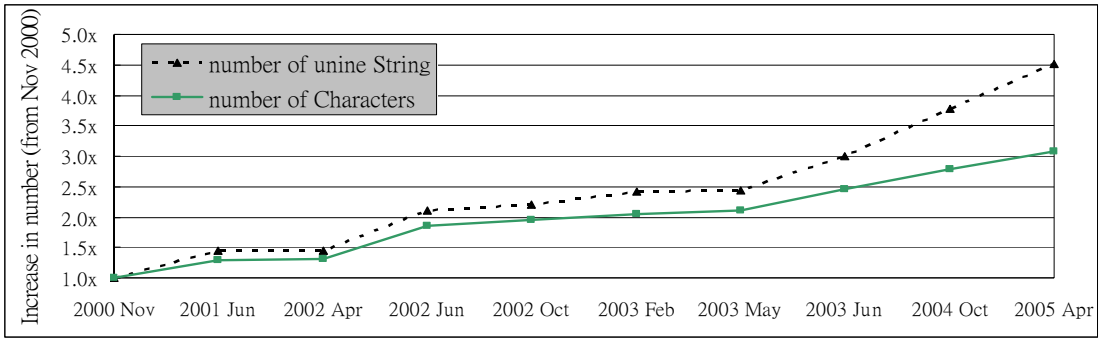

Figure 3-1. The growth of the Snort rule set over the last five years

3.1.2 Multi-event rule

For clarity, we define the Snort rule form below.

Definition 1:

First, we define the Snort rules as follows:

alert PROTOCOL SRC_ADDR SRC_PORT -> DES_ADDR DES_PORT(
msg: ……;
flow: ……;
condition1: ……; relation1;
condition2: ……; relation2;
condition3: ……; relation3; …
sid: ……; rev: ……;
)

Payload keywords ("*content*", "*uricontent*" and "*pcre*") in Snort rules are called "conditions". Others ("*depth*", "*offset'*, "*distance*" and "*within*") are called "relations".

Second, each (condition, relation) pair is an "event". The first (condition, relation) pair in the rule is called the "first event"; others are called "extended events". When a rule contains more than one event, it is called a "multi-event" rule. Table 3-2 shows an example of rules translated to (condition, relation) form.

Table 3-2. Translate several rules to (condition, relation) form

Rule	event 1 (first)	Event 2 (extend)	Event 3 (extend)	Event 4 (extend)
		Event 5 (extend)	Event 6 (extend)	Event 7 (extend)
1049	GET; depth:4	/../../../../../../../../../../ (URL content)		
1758	/b2/b2-include	B2inc	http|3A|//	
2009	error	|3A| no such repository	I HATE YOU	
1883	Uid=	|28|nobody|29|		
1884	Uid=	|28|web|29|		
1885	Uid=	|28|http|29|		
1989	MSG; depth:4	Content-Type|3a|	text/x-msmsgsinvite	Invitation-command|3A|
		CANCEL	Cancel-Code|3A|	REJECT
1988	MSG; depth:4	Content-Type|3a|	text/x-msmsgsinvite	Invitation-command|3A|
		Accept		
1986	MSG; depth:4	Content-Type|3a|	text/x-msmsgsinvite	Application-Name|3A|
		File transfer		
540	MSG; depth:4	Content-Type|3a|	text/plain	

The number of multi-event rules compared to the total number of rules has increased in later versions of Snort, as shown in Table 3-3. In the latest version of Snort (Snort 2.3, Apr, 2005), more than ¼ of all the rules are multi-event rules.

Table 3-3. The proportion of multi-event at several Snort rule sets

Ver	total rule numbers	multi-event rules	ratio	Max depth
2.0	2321	349	15.03%	4
2.2	2616	544	20.8%	6
2.3	2940	766	26%	7

3.2 Characteristic of Snort rule

When we examine the rules, we find several characteristics: flow, first event, group and interdependence. These characteristics can help us redesign the Snort system.

Flow:

The singularity-based NIDS employs a particular string to determine the purpose

of the attack and the same network applications always have a similar payload form. Thus, the singularity is similar among the same network applications.

We collect all the rules containing the content “YMSG” as shown in Table 3-4. This is an example means the content “YMSG” only checked on a TCP 5050 flow.

Table 3-4. All of the rules contain content “YMSG”

rule ID	Protocol	Flow	content1	content2
2450	TCP	5050 -> *	“YMSG”	"|00 01|"
2451	TCP	5050 -> *	“YMSG”	"|00|J"
2452	TCP	* -> 5050	“YMSG”	"|00 12|"
2453	TCP	5050 -> *	“YMSG”	"|00 18|"
2455	TCP	* -> 5050	“YMSG”	"|00 1D|"
2458	TCP	5050 -> *	“YMSG”	"|00 98|"
2459	TCP	* -> 5050	“YMSG”	"|00|P"

First event:

The Snort utilizes the first event to classify the applications. For example, the appearance of the content “YMSG” between 1~4 bytes represents the packet is “Yahoo IM” application. Some application examples are shown in Table 3-5.

Table 3-5. Application types and the corresponding first events

Application types	First events
NETBIOS SMB	"|00|****|FF|SMB"; depth:9
TFTP GET	"|00 01|"; depth:2
CHAT Yahoo IM	"YMSG"; depth:4
CHAT MSN message	"MSG "; depth:4

Group:

From above, we can state two facts:

1. The same network applications have similar payload behavior.

2. Snort uses the first event to classify packets.

Using this information, we attempt to use the first event to divide all the rules into several groups.

Definition 2:

1. All first events are given a unique number.
2. The “GroupNumber” of a rule is equal to the first event number.
3. Let “RuleGroup [i]” be the set of all rules whose GroupNumber is i.
4. Let “EventGroup [i]” be the set of all events in RuleGroup[i].
5. Let “EventGroupFirst [i]” be the first event of RuleGroup[i].
6. Let “EventGroupExtend [i]” be all the extended events of RuleGroup[i].”
7. Let “EventGroupSize [i]” be the size of EventGroup[i].
8. All elements of EventGroupExtend[i] are given unique “extend numbers”.
9. For each group, if the group size is smaller than four, it is called “SimpleGroup”, otherwise it is called “ComplexGroup”.

The search range of the original Detection Engine includes all events on the RTN list. This is different from the original method of using the first event to divide the rules into several independent RuleGroups (Figure 3-2). Since “Group” is a block, each group needs only to search under the corresponding RuleGroup. For this reason, narrowing the search range can greatly reduce the number of matches.

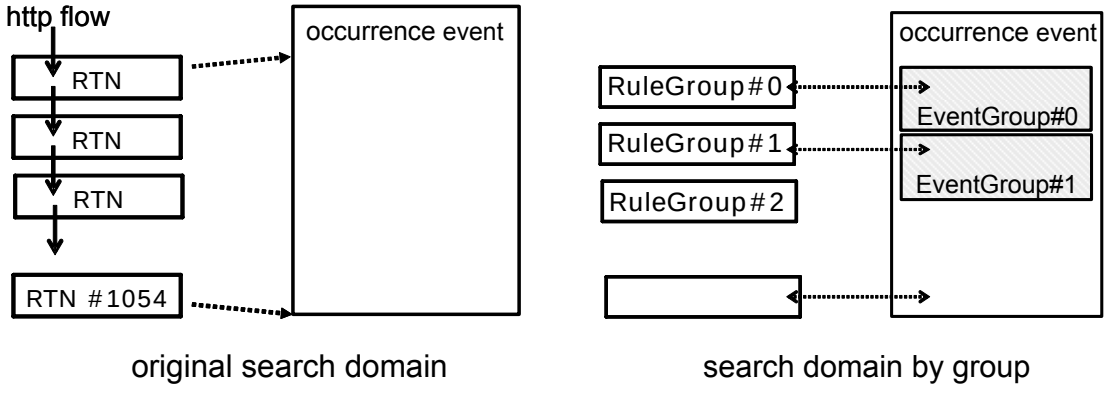

Figure 3-2. Original and group-based search domain

Interdependence:

Every rule contains a first event and extended events. For this reason, we can generalize to an important principle: the extended event only needs to process when

the first event appears in the same group. Based on this principle, the number of event correlation operations can be decreased.

3.3 Modified AC algorithm

Figure 3-3 shows that the length of most content is between four and sixteen bytes at the last version of the rule set. With short Snort content, there is a high probability that the pre-processor will produce pattern matching information. But usually, a single pattern match does not indicate real attack behavior [29].

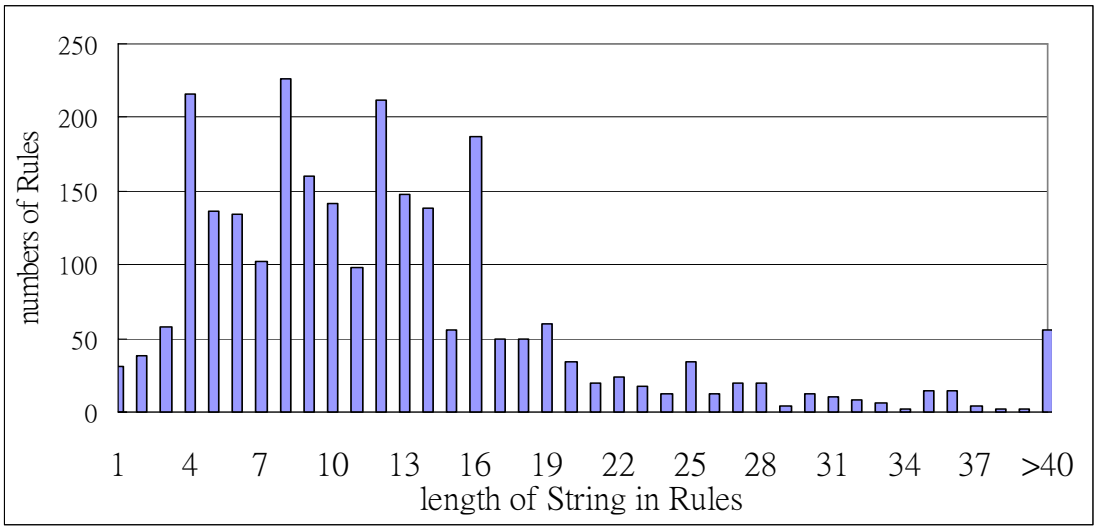

Figure 3-3. Length distribution of the unique strings in Snort 2.3

For example, “passwd” is an event that is used to identify the attack “FPT passwd retrieval attempt” under port 21 (rule ID: 356), but in other cases, the same content, “passwd”, also appears. If non-attack pattern match information can be reduced, the Detection Engine (post-processor) can substantially reduce overhead.

According to the Snort rule characteristic - flow, similar network applications have similar payload behavior. In an attempt to divide packet types into several different subsets, similar protocols are assigned to the same subset.

In special cases, some Snort rules must be checked under all types of flow.
For example :

```
alert any any any -> any any (
        msg: "Virus - Possible My Romeo Worm";…
        content: "Sorry... Hey you !"; …;
        sid:727; rev:6;
)
```

This means that the string "Sorry... Hey you!", which should be included in all subsets, must be checked for all protocols. A sub-AC tree is constructed for every subset and the rearranged AC tree is shown in Figure 3-4.

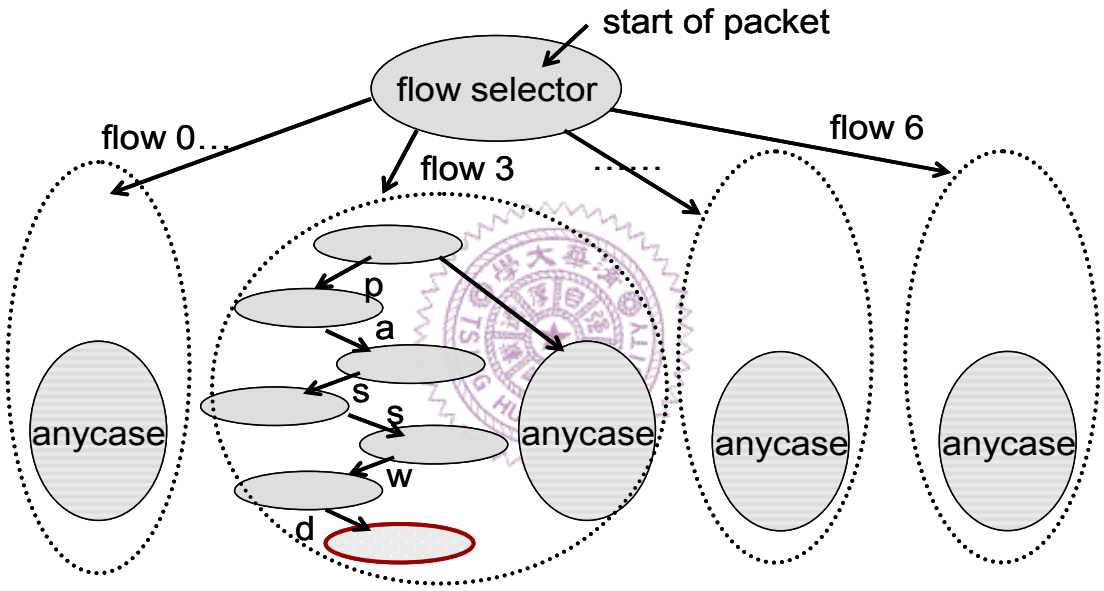

Figure 3-4. The architecture of modified AC tree

Fig 3-4 shows that “passwd” can be found when the packet’s source port is FTP. Actually, in the modified AC architecture, the content which the post-processor detects by correlation would never be a false positive.

After modifying the AC multi-pattern matching algorithm, only some memory space is required to filter unnecessary match information, substantially reducing the overhead of the post-processor.