A Stream Pattern Matching Method for Traffic Analysis*

Can Mo, Hui Li, and Hui Zhu

Lab of Computer Networks and Information Security,
Xidian University, Shaanxi 710071, P.R. China

Abstract. In this paper, we propose a stream pattern matching method that realizes a standard mechanism which combines different methods with complementary advantages. We define a specification of the stream pattern description, and parse it to the tree representation. Finally, the tree representation is transformed into the S-CG-NFA for recognition. This method provides a high level of recognition efficiency and accuracy.

Keywords: Traffic Recognition, Stream Pattern, Glushkov NFA.

1 Introduction

The most common traffic recognition method is the port-based method which maps port numbers to applications [1]. With the emergence of new applications, networks exceedingly carry more and more traffic that uses unpredicted port numbers which are dynamically allocated. As a consequence, the port-based method becomes insufficient and inaccurate in many cases.

The most accurate solution is payload-based method which searches the specific byte pattern-called signatures in all or part of the packets using deep packet inspection (DPI) technology[2,3], e.g. Web traffic contains the string 'GET'. However, there are many limits tied to this method. One of them is that some protocols are encrypted.

The statistics-based method utilizes the feature that different protocols correspond to different statistical characteristics [4]. For example, Web traffic is composed of short and small packets, while P2P traffic is usually composed of long and big packets. 289 kinds of statistical features of traffic or packets are presented in [5], including flow duration, payload size, packet inter-arrival time (IAT), and so on. However, this method can just coarsely classify the traffic into several classes, which limits the accuracy of traffic recognition, so this method can not be used alone.

In general, the currently available approaches mentioned above have respective strength and weakness, none of them performs well for all the different network data on the internet nowadays.

* Supported by "the Fundamental Research Funds for the Central Universities"(No.JY10000901018).

X. Lai et al. (Eds.): E-Forensics 2010, LNICST 56, pp. 131–140, 2011.

In this paper we propose a stream pattern matching method which implements a network traffic classification framework that is easy to update and configure. By the definition and specification design of the stream pattern, any kind of data stream with common features can be unambiguously described as a special stream pattern, according to a certain grammar and lexeme. Moreover the designed pattern combines different approaches at present, and can be flexibly written and expanded. In order to be easily understood by computer, a tree representation structure is obtained through a parser for the stream pattern. Then, for the recognition of network traffic, the parse tree is transformed into a Nondeterministic Finite Automata(NFA) with counters, called S-CG-NFA, and a stream pattern engine is built on it. The network traffic is sent to the stream pattern engine to get the matching result using the bit-parallel search algorithm.

The primary contribution of the stream pattern matching method is that three kinds of approaches (i.e, port-based method, payload-based method and statistics-based method) are combined in this method, and the efficiency of recognition is equivalent to a combined effect of these above approaches with complementary advantages, thus a more accurate recognition effect is achieved. Moreover, because of the standard syntax and unified way of parsing and identifying, the updating of the stream pattern is more simple than that of existing methods, so does the way of traffic recognition.

The remainder of this paper is organized as follows. Section 2 puts forward the definition and specification design of the stream pattern. The construction of a special stream parser based on the stream pattern is described in Section 3 and the generation of S-CG-NFA in Section 4. Experimental results can be found in Section 5. Section 6 presents the conclusion and some problems to be further solved.

2 The Design and Definition of the Stream Pattern

The stream pattern matching method proposed in our paper describes a network traffic classification framework that combines several classification approaches at present with complementary advantages and is easy to update and configure. The system framework is shown in Figure 1. First, the network traffic with certain features is described as the stream pattern. Second, a tree representation of the stream pattern is obtained by a stream parser. After that, the tree representation is transformed into S-CG-NFA to get the corresponding stream pattern matching engine. Any traffic to be recognized is first converted into characteristic flow through the collector, and then sent to the stream pattern engine. Finally, the matching result can be got from this engine. In this section, we will discuss the design and definition of the stream pattern.

The stream pattern is designed to be normative, and can unambiguously describe any protocol or behavior with certain characteristics based on the grammar and lexeme defined. Furthermore, for its good expansibility, the stream pattern can conveniently be added with any new characteristic.

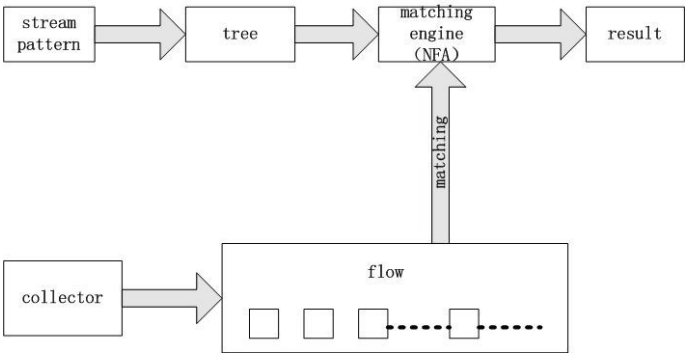

Fig. 1. system framework of the stream pattern matching

A stream pattern describes a whole data flow, and vice versa; that is, the stream pattern and the data flow are a one-to-one mapping. Here, the stream pattern is abstractly denoted as SM. Some formal definitions of the stream pattern are given in the following.

Definition 1. *A stream-character corresponds to a data packet in the data flow. It is the basic component of the stream pattern, which includes recognition features such as head information, payload information, statistical information, etc. The stream character is flexible to extend. The set of stream-character is denoted as $S\Sigma$, $s\omega \in S\Sigma$ denotes a formal stream-character, the empty stream-character is denoted as $s\epsilon$, the wildcards are denoted as "$\textcircled{sw}$".*

Definition 2. *A stream-operator describes the relationship between stream-characters. It is a basic component of the stream pattern including "$\textcircled{(}$", "$\textcircled{)}$", "$\textcircled{*}$", "$\textcircled{\oplus}$", "$\textcircled{?}$", "$\textcircled{\{\}}$", "$\textcircled{|}$", "$\textcircled{\cdot}$". The meaning of stream operators is described in Definition 4.*

Definition 3. *A stream pattern is a symbol sequence on the set of symbols $s\omega \in S\Sigma \bigcup \{s\epsilon, \textcircled{sw}, \textcircled{(}, \textcircled{)}, \textcircled{*}, \textcircled{\oplus}, \textcircled{?}, \textcircled{\{\}}, \textcircled{|}, \textcircled{\cdot}\}$ which is recursively defined according to a certain generating grammar. The generating grammar is as follows:*

$$
\begin{array}{ll}
SM \longrightarrow s\epsilon; & SM \longrightarrow s\omega \\
SM \longrightarrow \textcircled{(}SM\textcircled{)}; & SM \longrightarrow SM\textcircled{\cdot}SM \\
SM \longrightarrow SM\textcircled{|}SM; & SM \longrightarrow \textcircled{*} \\
SM \longrightarrow SM\textcircled{\oplus}; & SM \longrightarrow SM\textcircled{?} \\
SM \longrightarrow SM\textcircled{\{\}}. &
\end{array}
$$

Definition 4. *The network data flow represented by a stream pattern SM is described as $L(SM)$ and the meaning of each stream-operator is described as follows:*

For any $s\omega \in S\Sigma \bigcup s\epsilon$,

$$L(s\omega) = s\omega \tag{1}$$

$$L(SM_1 \bigoplus SM_2) = L(SM_1) \bigcup L(SM_2) \tag{2}$$

Equation 2 represents a union of the stream pattern SM_1 *and* SM_2.

$$L(SM_1 \bigodot SM_2) = L(SM_1) \bullet L(SM_2) \tag{3}$$

Equation 3 represents a concatenation of the stream pattern SM_1 *and* SM_2.

$$L(SM^{\circledast}) = \bigcup_{i \geq 0} L(SM)^i \tag{4}$$

Equation 4 represents a concatenation of zero or more sub-stream patterns represented by SM.

$$L(SM^{\oplus}) = \bigcup_{i \geq 1} L(SM)^i \tag{5}$$

Equation 5 represents a concatenation of one or more sub-stream patterns represented by SM.

$$L(SM^{?}) = L(SM) \bigcup L(s\epsilon) \tag{6}$$

Equation 6 represents a concatenation of zero or one sub-stream pattern represented by SM.

$$L(SM^{\{m,n\}}) = \bigcup_{m \leq i \leq n} L(SM)^i \tag{7}$$

Equation 7 represents that the sub-stream pattern is repeated a number of times specified by a lower and upper limit.

The stream-character contains three kinds of characteristics such as head information, payload information and statistics information. The characteristics used are shown in Table 1.

Any additional characteristic to the benefit of recognizing network traffic can be added to the stream pattern based on the specification defined above.

Table 1. Characteristic of stream-character

characteristic classes	feature items
head	source IP, destination IP, source port, destination port
payload	origin, offset, content
statistics	packet size, inter-arrival-time of packet, direction of packet

3 The Construction of the Parse Tree

After the design and definition of the stream pattern, we parse the stream pattern to obtain a tree representation, called parse tree that can be easily understood by computer to perform calculations. The parse tree corresponds to the stream pattern one-to-one: the leaves of the tree are labeled with stream-character, the intermediate nodes are labeled with the stream-operator, and recursively the sub tree corresponds to the sub stream pattern.

The grammar for the stream pattern is too complex for a lexical analyzer and too simple for a full bottom-up parser. Therefore, a special parser for the stream pattern is built, which is shown in Figure 2. Here "θ" represents an empty tree, ST represents an empty stack. The end of the stream is marked with ψ.

Parse(SM=$s\omega_1$, $s\omega_2$, ..., $s\omega_i$, ..., $s\omega_n$, $last$, ST)
    $\nu \longleftarrow \theta$
    While $SM_{last} \neq \psi$ Do
        If $SM_{last} \in S\Sigma$ OR $SM_{last} = s\epsilon$ Then
            $\nu_r \longleftarrow$Create a node with SM_{last}
            If $\nu \neq \theta$ Then
                $\nu \longleftarrow [\odot](\nu, \nu_r)$
            Else $\nu \longleftarrow \nu_r$
            $last \longleftarrow last + 1$
    Else If $SM_{last} = \textcircled{|}$ Then
        if $\nu = \theta$
            Return Error
        $(\nu_r, last) \longleftarrow$ Parse$(SM, last + 1, ST)$
        $\nu \longleftarrow [\textcircled{|}](\nu, \nu_r)$
    Else If $SM_{last} = \circledast$ Then
        $\nu \longleftarrow [\circledast](\nu)$
        $last \longleftarrow last + 1$
    Else If $SM_{last} = \oplus$ Then
        $\nu \longleftarrow [\oplus](\nu)$
        $last \longleftarrow last + 1$
    Else If $SM_{last} = \textcircled{?}$ Then
        $\nu \longleftarrow [\textcircled{?}](\nu)$
        $last \longleftarrow last + 1$
    Else If $SM_{last} = \{\}$ Then
        $\nu \longleftarrow [\{\}](\nu)$
        $last \longleftarrow last + 1$
    Else If $SM_{last} = \textcircled{(}$ Then
        PUSH(ST)
        $(\nu_r, last) \longleftarrow$ Parse$(SM, last + 1, ST)$
        If $\nu \neq \theta$ Then
            $\nu \longleftarrow [\odot](\nu, \nu_r)$
        Else

$\nu \longleftarrow \nu_r$
$last \longleftarrow last + 1$
Else If $SM_{last} = ⓙ$ Then
POP(ST)
Return(ν,$last$)
End of If
End of While
If !EMPTY(ST)
Return Error
Else Return(ν,$last$)

Fig. 2. The parse algorithm of the stream pattern

4 The Generation of S-CG-NFA

For recognition, the tree representation should be transformed into automata. Considering the features of the stream pattern and network traffic, a special automata for the stream pattern, called S-CG-NFA is presented which is based on Glushkov NFA [6,7] and extended with counters to better resolve numerical constraints. Automata with counter has been proposed in many papers, and has well resolved the problem of constrained repetitions [8,9,10,11,12,13]. Therefore, referred to the method presented in the reference [13], the construction of S-CG-NFA is given in the following.

For simplicity, we first give some statements to better resolve constrained repetitions. A sub-stream pattern of the form ⓘ is called an iterator. Each iterator c contains a lower limit as $lower(c)$, an upper limit as $upper(c)$ and a counter as cv(c). We denote by $iterator(x)$ the list of all the iterated sub stream patterns which contain stream-character x; we denote by $iterator(x, y)$ the list of all the iterated sub stream patterns which contain stream-character x, expect stream-character y. Several functions about iterators are defined as follows.

1. value_test(C): true if $lower(C) \leq cv(C) \leq upper(C)$, else false; check whether the value of cv(C) is between the lower limit and upper limit.
2. reset(C): $cv(C) = 1$; the counter of iterator C is reset to 1.
3. update(C): $cv(C)$++; the counter of iterator C is increased by 1.

Now, we give the construction of S-CG-NFA. S-CG-NFA is generated on the basis of the sets $First$, $Last$, $Empty$, $Follow$ and C. Here, the definitions of sets $First$, $Last$ and $Empty$ are the same as in the standard Glushkov construction, which will not be explained further. However, it is necessary to state that the set of C indicates all the iterators in the stream pattern, and the set $Follow$ being different from the standard set $Follow$ containing a $two-tuples(x, y)$, contains a $triple(x, y, c)$, where x and y are the positions of the stream-character in the stream pattern and c can be null or the iterator in the set C.

So the S-CG-NFA that represents the stream pattern is built in the following way.

$$\text{S-CG-NFA} = (Q_{SM} \bigcup \{q_0\}, S\Sigma^*, C, \delta_{SM}, q_0, F_{SM}) \tag{8}$$

In Equation (8), where

1. Q_{SM} is the set of states and the initial state is $q_0 = 0$;
2. $S\Sigma^*$ is the set of transition conditions, and is constituted with triple(*conid*, *sw*, *actid*). Among them, the element $sw \in S\Sigma$ is a stream-character, the element *conid* represents the set of conditional iterators and the element *actid* represents the set of responding iterators;
3. F_{SM} is the set of final states. For every element $x \in last$, if value_test(iterator(x)) = true, then $q_x \in F_{SM}$;
4. C is the set of all the iterators in the stream pattern;
5. δ_{SM} is the transition function of the automaton. $\delta_{SM} = (q_s, tc, \varphi, \pi, q_f)$; that is, for all $y \in first$, (0, (*null*, sw_y, *null*), true, Φ, y) $\in \delta_{SM}$; for all $x \in$Pos(SM) and (y, $SM^{'}$)$\in follow$, (x, (*conid*, sw_y, *actid*), φ, π, y) $\in \delta_{SM}$ if and only if φ=true. Among them, if $SM^{'} = null$, then *conid* = iterator(x, y); *actid* = *null*, φ= value_test(*conid*), π = reset(*conid*); otherwise, *conid* = iterator(x, $SM^{'}$); *actid* = $SM^{'}$, φ = value_test(*conid*), π = reset(*conid*)$\bigcup$ update(*actid*).

So far, the whole construction process of S-CG-NFA has been described. Considering the complexity of S-CG-NFA, here we use the one-pass scan algorithm and the bit-parallel search algorithm to recognize the network traffic data.

5 Experimental Evaluation

In the above section, we give the design and realization of the stream pattern matching engine which is implemented in C/C++ development environment and on the basis of function library LibXML2 [14]. In this section, we briefly present an experimental evaluation on the effect of the stream pattern matching technology. We take the HTTP protocol for example and give two kinds of stream patterns describing HTTP. Stream pattern 1 describes HTTP just contains port information which is shown in Figure 3. Stream pattern 2 describes HTTP combined with port information and payload information which is shown in Figure 4.

The two stream patterns are applied in four traces to separately get the total number of HTTP flows recognized. The four traces are from DARPA data sets [15](1998, Tuesday in the third week, 82.9M; 1998, Wednesday in the fourth week, 76.6M; 1998, Friday in the fourth week, 76.1M; 1998, Wednesday in the fifth week, 93.5M). A list file records the number of http flows got by port-based method in each trace which is selected as the base of comparison. The recognition result is shown in Table 2, where the first column corresponds to the number of http flows recorded in the list file, the second column corresponds to the number of http flows recognized by stream pattern 1 and the third column corresponds to the number of http flows recognized by stream pattern 2.

```
<mode>
     <element type_id=‘‘word">
         <head>
            <dport>80</dport>
          </head>
          <content>NULL</content>
        <statistic>
            <dir>0</dir>
        </statistic>
</mode>
```

Fig. 3. Stream pattern 1 for HTTP

```
<mode>
    <element type_id=‘‘word">
        <head>
            <dport>80</dport>
        </head>
        <content>
            <within>100</within>
            <offset>0</offset>
            <con>GET</con>
        </content>
        <statistic>
            <dir>0</dir>
        </statistic>
    </element>
    <element type_id=‘‘word">
        <head>
            <sport>80</sport>
        </head>
        <content>
            <within>100</within>
            <offset>0</offset>
            <con>HTTP</con>
         </content>
         <statistic>
            <dir>1</dir>
         </statistic>
    </element>
</mode>
```

Fig. 4. Stream pattern 2 for HTTP

Table 2. recognition result for HTTP

Trace file	list file	stream pattern 1	stream pattern 2
Trace 1	5016	5016	5016
Trace 2	4694	4694	766
Trace 3	2233	2233	158
Trace 4	4833	4833	67

Table 2 shows that the stream pattern matching engine can be reduced to port-based method using stream pattern 1 to achieve 100% recognition rate, that is the stream pattern matching technology can have the same effect as the port-based method. However, due to the existence of incomplete data flows which just contain handshake information and have no transmission content, the number of flows recognized by stream pattern 2 is less than stream pattern 1, since some fake http flows are removed. So at some point, the recognition accuracy of stream pattern 2 which combines both port-based method and payload-based method is higher.

From the above, it is clear that the stream pattern matching technology not only can combine different methods with complementary advantages, but also is easy to expand.

6 Conclusion and Future Work

In this paper, we have introduced a stream pattern matching technology, which provides a recognition framework that combines three kinds of recognition methods with complementary advantages. It is easy to configure and update. We provide a formal definition of the stream pattern, and then convert the text form of the stream pattern into the tree representation. Finally, we transform the parser tree to the S-CG-NFA, a special automata for the stream pattern to generate the stream pattern matching engine. We performed a system test and the test result shows the effectiveness of the stream pattern matching engine.

However, there are some aspects that need further effort.

1. The generation of the stream pattern: the stream pattern is manually written after the manual analysis of network data or the reference to the existing literature, so the validity and reliability of the generation way of the stream pattern are challenging and need to be improved. And also the automatic generation of the stream pattern is a future direction.
2. The speed of matching: Since different protocols correspond to different matching engines and any network data that needs to be recognized should be sent to every engine, so the processing speed of matching engine is highly demanded. Therefore, the study of parallel processing is a vital task.

References

1. IANA, http://www.iana.org/assignments/port-numbers
2. Kang, H.-J., Kim, M.-S., Hong, J.W.-K.: A method on multimedia service traffic monitoring and analysis. In: Brunner, M., Keller, A. (eds.) DSOM 2003. LNCS, vol. 2867, pp. 93–105. Springer, Heidelberg (2003)
3. Levandoski, J., Sommer, E., Strait, M.: Application Layer Packet Classifier for Linux[CP/OL] (2006), http://l7-filter.sourceforge.net/
4. Zuev, D., Moore, A.W.: Traffic classification using a statistical approach. In: Dovrolis, C. (ed.) PAM 2005. LNCS, vol. 3431, pp. 321–324. Springer, Heidelberg (2005)
5. Moore A.W., Zuev D., Crogan M.: Discriminators for use in flow based classification. Department of Computer Science, Queen Mary, University of London (2005)
6. Berry, G., Sethi, R.: From regular expression to deterministic automata. Theoretical Computer Science 48(1), 117–126 (1986)
7. Chang, C.H., Paige, R.: From regular expression to DFA's using NFA's. In: Proceedings of the 3rd Annual Symposium on Combinatorial Pattern Matching. LNCS, vol. 664, pp. 90–110. Springer, Heidelberg (1992)
8. Kilpeläinen, P., Tuhkanen, R.: Regular Expressions with Numerical Occurrence Indicators-preliminary results. In: Proceedings of the Eighth Symposium on Programming Languages and Software Tools, SPLST 2003, Kuopio, Finland, pp. 163–173 (2003)
9. Kilpeläinen, P., Tuhkanen, R.: One-unambiguity of regular expressions with numeric occurrence indicators. Inf. Comput 205(6), 890–916 (2007)
10. Becchi, M., Crowley, P.: Extending Finite Automata to Efficiently Match Perl-Compatible Regular Expressions. In: Proceedings of the 2008 ACM Conference on Emerging Network Experiment and Technology, CoNEXT 2008, Madrid, Spain, vol. 25 (2008)
11. Becchi, M., Crowley, P.: A Hybrid Finite Automaton for Practical Deep Packet Inspection. In: ACM CoNEXT 2007, New York, NY, USA, pp. 1–12 (2007)
12. Yun, S., Lee, K.: Regular Expression Pattern Matching Supporting Constrained Repetitions. In: Proceedings of Reconfigurable Computing: Architectures, Tools and Applications, 5th International Workshop, Karlsruhe, Germany, pp. 300–305 (2009)
13. Gelade, W., Gyssens, M., Martens, W.: Regular Expressions with Counting: Weak versus Strong Determinism. In: Proceedings of Mathematical Foundations of Computer Science 2009, 34th International Symposium, Novy Smokovec, High Tatras, Slovakia, pp. 369–381 (2009)
14. LIBXML, http://www.xmlsoft.org/
15. DARPA, http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html