Optimization of Pattern Matching Algorithm for Memory Based Architecture

Cheng-Hung Lin, Yu-Tang Tai, and Shih-Chieh Chang

National Tsing Hua University, Taiwan, R.O.C
Outline

- Memory architecture for string matching
- Basic idea
- Novel Algorithm for memory architecture
- Experimental results and conclusions
Introduction

- Network Intrusion Detection System is used to detect network attacks by identifying attack patterns.

- Software-only approaches can no longer meet the high throughput of today’s networking.

- Hardware approaches for acceleration.
  - Logic architecture
  - Memory architecture
Advantage of Memory Architecture

- The memory architecture has attracted a lot of attention because of its easy re-configurability and scalability.


Memory Architecture

Attack Patterns

“bcdf”
“pcdg”

Memory

Current state
Decoded
256:1 MUX

Input

match vector

FSM

0 1 2 3 4
5 6 7 8

0123456789

NS1 NS2 ...... NS256 MV
<8> <8> ...... <16>

<8> <8> ...... <8> <16>

<N> <N> ...... <N> <N>

<8> <8> ...... <8> <8>

<8> <8> ...... <8> <8>
Due to the increasing number of attacks, the required memory increases tremendously:
- The performance, cost, and power consumption are related to the memory size.
- Reducing the memory size has become imperative.
Outline

- Memory architecture for string matching
- Basic idea
- Novel algorithm for memory architecture
- Experimental results and Conclusions
Aho-Corasick (AC) algorithm can reduce large number of state transitions and memory size.

- Solid line represents valid transitions.
- Dotted line represents failure transitions.
- Introduce the failure transition to reduce the outgoing transitions.

AC state machine of “bcdf” and “pcdg”
Observation

- Many string patterns are similar because of common sub-strings
- The similarity does not lead to a small state machine.

“bcdf”
“pcdg”

AC state machine
The merg_fsm is a different machine
- smaller number of states and transitions.
- smaller memory in memory architecture.
Problem of merg_FSM

- Directly merging similar states results in an erroneous state machine.

input stream = \{p, c, d, f\}

AC state machine

merg_FSM
Outline

- Memory architecture for string matching
- Basic Idea
- Novel Algorithm for memory architecture
- Experimental results and Conclusions
State Traversal Mechanism

- Store **merg_FSM** table in memory
- **State traversal mechanism** is used to memorize the precedent state and differentiate merged states.

AC state machine

![Diagram of AC state machine and merg_FSM state traversal mechanism]
New State Information

- AC state machine stores *match vector*.
- New state machine stores
  - *PathVec* stores path information.
  - *IfFinal* indicates whether the state is a final state.
Pseudo-Equivalent States

- Definition: Two states are *pseudo-equivalent* if they have
  - identical input transitions
  - identical failure transitions
  - identical ifFinal
  - but different next states.
Merge Pseudo-Equivalent States

Pseudo-equivalent states are merged.

PathVec and ifFinal are updated by a union of merged states.
PreReg traces the precedent pathVec in each state.

input stream: \{p, c, d, f\}
Outline

- Memory architecture for string matching
- Basic Idea
- Novel algorithm for memory architecture
- Experimental results and Conclusions
Experiment I

- Perform experiments on Snort rule sets.
- Compare our approach with the Aho-Corasick algorithm.

## Compare with Traditional AC

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of trans.</td>
<td># of states</td>
<td>Memory (bytes)</td>
<td># of trans.</td>
<td># of states</td>
</tr>
<tr>
<td>Oracle</td>
<td>138</td>
<td>2,180</td>
<td>2,185</td>
<td>880,009</td>
<td>1,389</td>
</tr>
<tr>
<td>Sql</td>
<td>44</td>
<td>421</td>
<td>422</td>
<td>129,290</td>
<td>321</td>
</tr>
<tr>
<td>Backdoor</td>
<td>57</td>
<td>563</td>
<td>565</td>
<td>191,253</td>
<td>523</td>
</tr>
<tr>
<td>Web-iis</td>
<td>113</td>
<td>1,533</td>
<td>1,537</td>
<td>569,651</td>
<td>1,273</td>
</tr>
<tr>
<td>Web-php</td>
<td>115</td>
<td>1,670</td>
<td>1,675</td>
<td>620,797</td>
<td>1,295</td>
</tr>
<tr>
<td>Web-misc</td>
<td>310</td>
<td>3,576</td>
<td>3,587</td>
<td>1,444,664</td>
<td>3,031</td>
</tr>
<tr>
<td>Web-cgi</td>
<td>347</td>
<td>3,407</td>
<td>3,419</td>
<td>1,377,002</td>
<td>2,672</td>
</tr>
<tr>
<td>Total rules</td>
<td>1,595</td>
<td>17,472</td>
<td>17,522</td>
<td>8,745,668</td>
<td>14,704</td>
</tr>
<tr>
<td>Ratio</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>84%</td>
<td>76%</td>
</tr>
</tbody>
</table>
Experiment II

- Enhance the bit-split algorithm with our method
  - The results are compared with the original bit-split algorithm.

## Compare with Traditional Bit-Split

<table>
<thead>
<tr>
<th>Rule Sets</th>
<th># of patterns</th>
<th># of char.</th>
<th>Bit-split [8]</th>
<th>Bit-split + Our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of trans.</td>
<td># of states</td>
<td>Memory (bytes)</td>
<td># of trans.</td>
</tr>
<tr>
<td>Oracle</td>
<td>138</td>
<td>4,674</td>
<td>6,645</td>
<td>6,665</td>
</tr>
<tr>
<td>Sql</td>
<td>44</td>
<td>1,089</td>
<td>1,211</td>
<td>1,215</td>
</tr>
<tr>
<td>Backdoor</td>
<td>57</td>
<td>599</td>
<td>1,697</td>
<td>1,705</td>
</tr>
<tr>
<td>Web-iis</td>
<td>113</td>
<td>2,047</td>
<td>4,869</td>
<td>4,885</td>
</tr>
<tr>
<td>Web-php</td>
<td>115</td>
<td>2,455</td>
<td>4,991</td>
<td>5,011</td>
</tr>
<tr>
<td>Web-misc</td>
<td>310</td>
<td>4,711</td>
<td>10,959</td>
<td>11,003</td>
</tr>
<tr>
<td>Web-cgi</td>
<td>347</td>
<td>5,339</td>
<td>9,901</td>
<td>9,949</td>
</tr>
<tr>
<td>Total rules</td>
<td>1,595</td>
<td>20,921</td>
<td>53,930</td>
<td>54,130</td>
</tr>
<tr>
<td><strong>Ratio</strong></td>
<td></td>
<td></td>
<td><strong>1</strong></td>
<td><strong>1</strong></td>
</tr>
</tbody>
</table>
Conclusion

- Provide a concept of merging pseudo-equivalent states to reduce the number of states and transitions.
- Propose a state traversal mechanism working with the merg_FSM without false positive matching results.
- Experimental results demonstrate a significant reduction in memory requirement.
Thank You!
Backup
Merging disorder sections of pseudo-equivalent states creates cycle problem.
Cycle Problem

- For example, the input string “abcdebcdedef” will be mistaken as a match of the pattern “abcdef.”
Construction of State Traversal Machine

Construction of the state traversal machine consists of two steps

- Step1: Construct valid transitions, failure transitions, pathVec, and ifFinal function.
- Step2: Merge the pseudo-equivalent states.
Example

- Consider three patterns “abcdef”, “apcdeg”, “awcdeh”.

16 states
Merging Pseudo-equivalent States

- merging the failure transitions
- performing the union on the pathVec of the merged states
Merging Pseudo-equivalent States
Merging Pseudo-equivalent States

10 states

Diagram showing states and transitions with binary codes and labels.
State Traversal Algorithm

**Algorithm:** State traversal pattern matching algorithm

**Input:** A text string \( x = a_1a_2 \ldots a_n \) where each \( a_i \) is an input symbol and a state traversal machine \( M \) with valid transition function \( g \), failure transition function \( f \), path function \( pathVec \) and final function \( ifFinal \).

**Output:** Locations at which keywords occur in \( x \).

**Method:**

begin
  \( state \leftarrow 0 \)
  \( preReg \leftarrow 1 \ldots 1 \) // all bits are initiated to 1.
  for \( i \leftarrow \text{until } n \) do
    begin
      \( preReg = preReg \land pathVec(state) \)
      while \( g(state, a_i) == \text{fail } \| \ preReg == 0 \) do
        begin
          \( state \leftarrow f(state) \)
          \( preReg \leftarrow 1 \ldots 1 \)
        end
      state \leftarrow g(state, a_i)
      if \( ifFinal(state) = 1 \) then
        begin
          print \( i \)
          print \( preReg \)
        end
    end
end