A Framework for Rule Processing in Reconfigurable Network Systems

Michael Attig and John Lockwood

Washington University in Saint Louis
Applied Research Laboratory
Department of Computer Science and Engineering
May 1, 2005

Outline

• Overview
• Background
• Architecture
• Results
• Summary
Rule Processing Overview

- Rule processing & intrusion detection
- TCP Flow Processing
- Header Processing
- Payload Scanning

```
alert tcp any 110 ⇒ any any (msg:“Virus - Possible MyRomeo Worm”;
    flow:established; content:”myromeo.exe”; nocase; classtype:misc-
    activity; sid:723; rev:6;)
```

- Snort Rules (version 2.2 Sept 2004)
  - 2464 Rules
  - 292 Headers
  - 2107 Signatures
  - 233 Regular Expressions

Rule Characteristics

- 2464 unique rules
- 292 unique header rules
  - 168 are “header-only”
- 2107 unique signatures
  - 97% less than 32 bytes
  - Spread across 2296 of rules
- 233 regular expressions
  - Snort rules always contain static signature also

- Few signatures associated with many rules
  - 83% found in single rule
  - Only 18 associated with more than 10 rules
- 10 header rules can match at once (pessimistic)
Fully Functional Rule Processing

String Matching
- Comparators [Soundis fccm'04]
- Bloom Filters [Attili et al.'04]
- DFAs [Moscola fccm'03]

Partitioning [Baker fccm'04]
- TCAMs [Yu hoti'04]
- BV-TCAM [Song fpga'05]

Header Classification
- NFAs [Clark fccm'04]

TCP Flow Reconstruction [Schuehler fpl'04]

Pipelining [Cho fccm'04]

Rule Processing Framework Overview

Order TCP flows for inspection by payload scanner(s)

h header module determines header

p payload module searches for static signatures and regular expressions

Rule processor uses header and payload match criteria to determine rule matches

Intrusion Detection System

Focus of this Talk

TCP Flow Assembler

Local Area Network

Internet

Rule Processor

Intrusion Detection System

Local Area Network

Internet

Intrusion Detection System

Rule Processor

Focus of this Talk
Example

R1: Alert tcp any 80 → any 125  
(content:"string1"; content:"string2";)

Algorithmically:

R1: H1 ∧ C1 ∧ C2
Rule Processing Example

TCP Flow Assembler

Header Processor

Payload Scanner

Rule Processor

Rule Processing Framework – FCCM 2005

Rule Processor

FPGA

Software Communication Wrapper

Control FSM

Context Storage

Rule Generator

Alert Generator

Matching Criteria Communication Wrapper

ID Debt

Reset

Context Storage

+1

Rule Processing Framework – FCCM 2005
Implementation Environment

- **Xilinx Virtex 2000E FPGA**
  - 12% LUTs
  - 25% Slices
  - 93% of Block RAMs
  - 80.6 MHz
- **Stacked configuration allows chaining processing**

Worst Case Throughput

- Worst case when signature associated with many rules detected
- Only 18 signatures in more than 10 rules
- Worst case signature
  - |00 00 00 00| in 135 rules
- **Scenario:**
  - Back-to-back 44 byte TCP packets from different flows and |00 00 00 00| as payload
  - Worst case assumes 7 million attack packets per second
Intrusion Detection of WashU’s Backbone Network

Matching 4 Byte Signatures
Matching 12+ Byte Signatures

- Observe ~10,000 total string matches per second on WashU’s backbone network (~250-300 Mbps)
- Scaling to 2.5 Gbps, only ~100,000 string matches per second

Next Generation FPGA Projections

- More block RAM
- Faster place & route
- Parallel copies of pipeline
  - Multiple IDs per clock cycle
- QDR SRAMs
- 6x improvement to throughput
### Related Work

<table>
<thead>
<tr>
<th>Function</th>
<th>Group and Component</th>
<th>Device</th>
<th>Logic Cells</th>
<th>Throughput (Gbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow</td>
<td>GaTech Stream Assembler</td>
<td>Virtex 1000</td>
<td>876 (10%)</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>WashU TCP Processor</td>
<td>Virtex4 140</td>
<td>22,100 (35%)</td>
<td>10.3</td>
</tr>
<tr>
<td>Header Processing</td>
<td>WashU BV-TCAM</td>
<td>Virtex4 100</td>
<td>4,200 (10%)</td>
<td>10</td>
</tr>
<tr>
<td>Payload</td>
<td>Crete Pre-decoded CAMs</td>
<td>Virtex2-6000</td>
<td>64,268 (95%)</td>
<td>9.7</td>
</tr>
<tr>
<td>Scanning</td>
<td>GaTech Decoder Trees</td>
<td>Virtex2-8000</td>
<td>54,890 (81%)</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Tokyo Trie-based Hash</td>
<td>Virtex2-6000</td>
<td>2,365 (7%)</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>UCLA Packet Filters</td>
<td>Spartan 3 2000</td>
<td>15,202 (37%)</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>USC Partitioning</td>
<td>Virtex2 Pro</td>
<td>15,010 (15%)</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>WashU Bloom Filters</td>
<td>Virtex4 100</td>
<td>35,850 (85%)</td>
<td>20.4</td>
</tr>
<tr>
<td>Correlation</td>
<td>WashU Rule Processor</td>
<td>Virtex4 100</td>
<td>40,200 (95%)</td>
<td>15.9</td>
</tr>
</tbody>
</table>

### Contributions

- **Development of large-scale Rule Processing Framework**
  - Bridge between component processing and rule processing
  - Supports up to 32,768 rules
- **Rule processing framework capable of 2.5 Gbps throughput on FPX**
  - Projected to 15.9 on latest Virtex 4
- **Rule processor operated on TCP flows**
  - Context information stored for over 2 million simultaneous flows
Acknowledgments

- Research Sponsors
  - Global Velocity
  - Boeing

- ARL Faculty & Students

http://arl.wustl.edu/projects/fpx/reconfig.htm

Questions?
Communication Wrapper Interface

Between Devices

Between Software/Hardware

Example

R1: Alert tcp any 80 \(\rightarrow\) any 125
(content:"string1"; content:"string2";)

R2: Alert tcp any 8080 \(\rightarrow\) any 1024
(content:"string1";)

R1: H1 \& C1 \& C2
R2: H2 \& C1
Adding Modules

- Accept and act upon IP packets
- Communicate match criteria using communication wrapper interface
  - Provide deterministic interfaces
  - Abstract transport protocol
- Software Configuration using communication wrapper
- Represent matching criteria as ID numbers

- Allows combination of techniques
  - Take advantage of best characteristics
    - General classifiers vs. field-specific headers
    - Static strings vs. regular expressions
Evaluation

- Recall Rule IDs are inserted into pipeline based on matching signatures

```
<table>
<thead>
<tr>
<th>Rule Look-ups per Second</th>
<th>Throughput (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.E+00</td>
<td>1.E+00</td>
</tr>
<tr>
<td>1.E+02</td>
<td>1.E+02</td>
</tr>
<tr>
<td>1.E+04</td>
<td>1.E+04</td>
</tr>
<tr>
<td>1.E+06</td>
<td>1.E+06</td>
</tr>
<tr>
<td>1.E+08</td>
<td>1.E+08</td>
</tr>
</tbody>
</table>
```

- 80 Million rule IDs per second

Additional Rules Supported

- Virtex 2
  - 120 of 144 Block RAMs (18 Kbits each)
  - 2 copies of pipeline
    - 10 BRAM in stage 2
    - 10 BRAM in stage 5
    - 40 BRAM in stage 6
  - 184,320 rules supported

- Virtex 4
  - 216 of 240 Block RAMs (18 Kbits each)
  - 4 copies of pipeline
    - 9 BRAM in stage 2
    - 9 BRAM in stage 5
    - 36 BRAM in stage 6
  - 165,888 rules supported