A Configurable-Hardware Document-Similarity Classifier to Detect Web Attacks

Craig Ulmer
cdulmer@sandia.gov

April 20, 2010

Craig Ulmer
Maya Gokhale
Sandia National Laboratories, California
Lawrence Livermore National Laboratory
Overview

- Network security is challenging, especially at link speed
  - FPGAs offer convenient means of brute-force pattern matching
  - Attackers game network intrusion detection systems
- Network researchers: machine learning for better classification
  - Document Similarity via TFIDF and Cosine Similarity
  - Found >94% accuracy in HTTP attack classification
  - But, slow and utilized 46MB of dictionary data
- Adapt document similarity to an embedded form
  - Simplifications, dictionary reductions, parallel Bloom filters
  - Tools to automatically generate FPGA hardware
  - Achieve 94% accuracy with 128KB dictionary at GigE rates
Outline

• Introduction
  – Discovery challenge and LLNL approach

• Adapting to embedded hardware
  – Algorithm modifications
  – Data modifications

• Implementation details
  – Core design
  – Tool flow
  – Performance and resource utilization

• Future work
ECML/PKDD 2007 Discovery Challenge

• HTTP Traffic Classification
  – Apply machine learning to identify malicious activity in HTTP

• Hand-labeled datasets of HTTP flows
  – Training: 50K inputs, 30% attacks
  – Testing: 70K inputs, 40% attacks
  – 7 Attack Types XSS, SQL/LDAP/XPATH injection, path traversal, command execution, and SSI

Flow Example

GET /eH/first_str/2hFnull6/oixsotcwrsamgit2/38PrR_Lkmnoz0.htm
Host: www.a215Een.st:15
Connection: close
Accept: */*
Accept-Charset: *;q=0.4
Accept-Encoding: *
Accept-Language: boHEor-sen0, gte-htmse4 oS, 3TeoUsHn-asrao;q=0.2, paly-wreihi, 78iiqths-ar;q=0.3
Cache-Control: no-store
Client-ip: 200.91.18.159
Cookie: ucly2kleicl=%3C%21---%23odbc++++++++++++++connect%3D%22at8h%2CHteil%2CeHnNa%22++++statement%3D%22drop+table+elkbO…
Prior LLNL Work

- Brian Gallagher and Tina Eliassi-Rad
- Document similarity: vector approach
  - Tokenize input
  - Assign weights to tokens via TFIDF
  - Cosine similarity for vector comparison
- Relies on a data dictionary
  - Generate term statistics during training
  - Reference statistics at runtime
  - Each term: IDF value and C weights

<table>
<thead>
<tr>
<th>Term</th>
<th>IDF</th>
<th>CValid</th>
<th>CA1</th>
<th>CA2</th>
<th>CA3</th>
<th>CA4</th>
<th>CA5</th>
<th>CA6</th>
<th>CA7</th>
</tr>
</thead>
<tbody>
<tr>
<td>odbc</td>
<td>2.079</td>
<td>0.0002</td>
<td>0</td>
<td>0</td>
<td>0.0134</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>statement</td>
<td>2.079</td>
<td>0.0013</td>
<td>0</td>
<td>0</td>
<td>0.0134</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>--</td>
<td>0.988</td>
<td>0</td>
<td>0</td>
<td>0.011</td>
<td>0.0126</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Original Accuracy

![Graph showing accuracy vs total dictionary terms]

- **Training Dataset**
- **Testing Dataset**

Accuracy (Percent)

Total Dictionary Terms
Outline

• Introduction
  – Discovery challenge and LLNL approach

• Adapting to embedded hardware
  – Algorithm modifications
  – Data modifications

• Implementation details
  – Core design
  – Tool flow
  – Performance and resource utilization

• Future work
Hardware Adaptation Challenges

- **Computation**
  - Blocking form
  - Floating point math
  - Divide and square root operations

- **Dictionary: 46MB, 1.8M terms**
  - Large storage
  - Lookup overhead

- **Path for converting to hardware**
  - Build the hardware design once
  - Automatically update with configuration data
Modification 1: Computation Adjustments

\[
\text{score[classifier]} = \frac{\sum_{t \in \mathcal{R}} \left( \frac{\text{count}[t]}{\# \text{input terms}} \cdot \text{idf}[t] \right) \cdot \text{tfidf[classifier][t]}}{\sqrt{\sum_{t \in a} \left( \frac{\text{count}[t]}{\# \text{input terms}} \cdot \text{idf}[t] \right)^2} \cdot \text{tfidfMagnitude[classifier]}}
\]
Modification 1: Computation Adjustments

\[
\text{score}[\text{classifier}] = \frac{\sum_{t \in \text{AnR}} \left( \frac{\text{count}[t]}{\# \text{input terms}} \cdot \text{idf}[t] \right) \cdot \text{tfidf}[\text{classifier}][t]}{\sqrt{\sum_{t \in \text{AnR}} \left( \frac{\text{count}[t]}{\# \text{input terms}} \cdot \text{idf}[t] \right)^2 \cdot \text{tfidfMagnitude}[\text{classifier}]]}}
\]
Modification 1: Computation Adjustments

\[ \text{score}[\text{classifier}] = \frac{\sum_{t \in \text{term}} \left( \frac{\text{count}[t]}{\text{# input terms}} \right) \cdot \text{idf}[t] \cdot \text{tfidf}[^{\text{classifier}}][t]}{\sqrt{\sum_{t \in \text{a}} \left( \frac{\text{count}[t]}{\text{# input terms}} \cdot \text{idf}[t] \right)^2} \cdot \text{tfidfMagnitude[^{\text{classifier}}]}} \]
Modification 1: Computation Adjustments

\[
\text{score}[\text{classifier}] = \frac{\sum_{t \in \text{Input}} \frac{\text{count}[t]}{\# \text{input terms}} \cdot \text{idf}[t] \cdot t\text{fidf}[\text{classifier}][t]}{\sqrt{\sum_{t \in \text{Input}} \left(\frac{\text{count}[t]}{\# \text{input terms}} \cdot \text{idf}[t]\right)^2 \cdot \text{tfidfMagnitude}[\text{classifier}]}}
\]
Modification 1: Computation Adjustments

\[
\text{score}[\text{classifier}] = \frac{\sum_{t \in \text{terms}} \frac{\text{count}[t]}{\# \text{input terms}} \cdot \text{idf}[t] \cdot \text{tfidf}[\text{classifier}][t]}{\sqrt{\sum_{t \in \text{terms}} \left( \frac{\text{count}[t]}{\# \text{input terms}} \cdot \text{idf}[t] \right)^2} \cdot \text{tfidfMagnitude}[\text{classifier}]} 
\]
Modification 1: Computation Adjustments

\[
\text{score}[\text{classifier}] = \sum_{t \in \text{dictionary}} \frac{\text{count}[t]}{\# \text{input terms}} \cdot \text{idf}[t] \cdot \text{tfidf}[\text{classifier}][t]
\]

- Count each term in Input
- Lookup IDF for term in dictionary
- Lookup weight for term in dictionary
- Adjust based on weight of classifier
- Scale based on TF-IDFs found by ALL classifiers
Modification 1: Computation Adjustments

\[
\text{score}[\text{classifier}] = \frac{\sum_{t \in \text{Input terms}} \frac{\text{count}[t]}{\# \text{Input terms}} \cdot \text{idf}[t] \cdot \text{tfidf}[\text{classifier}][t]}{\sqrt{\sum_{t \in \text{Input terms}} \left( \frac{\text{count}[t]}{\# \text{Input terms}} \cdot \text{idf}[t] \right)^2}} \cdot \text{tfidfMagnitude}[\text{classifier}]
\]
Modification 1: Impact on Accuracy

[Graph showing the effect of total dictionary terms on accuracy for both training and testing datasets, with lines for original and streaming methods.]
Dictionary Observations

- Many terms in the dictionary
  - 1.8M terms (46MB text, 128MB data)
  - Many terms are junk (“rv:0.7.8”), but they also get very low weight

- Data values are not very diverse
  - Total unique values is < 2% of population
  - Eg: OSC Classifier has 102K terms, but only 415 unique weights
Modification 2: Truncate Dictionary

![Graph showing accuracy over total dictionary terms for different training and testing scenarios.](image)
Modification 3: Re-Quantize Data Values

Log-Histogram

Data Values

Occurrences

0.0 0.1 0.2 0.3 0.4 0.5 0.6

0.0 1.0 10.0 100.0 1000.0

0.0 10.0 100.0 1000.0

Sandia National Laboratories
Modification 4: Map to Q Bloom Filters

Bloom Filters

0.001

0.11

0.21

0.47
End Impact on Accuracy

Our Choice:
- 8 quantization levels/classifier
- 4K total terms
Outline

• **Introduction**
  – Discovery challenge and LLNL approach

• **Adapting to embedded hardware**
  – Algorithm modifications
  – Data modifications

• **Implementation details**
  – Core design
  – Tool flow
  – Performance and resource utilization

• **Future work**
Core Architecture

HTTP Input Byte Stream

Input Stream Tokenizer

H Hash Functions

IDF Lookup

Category 0: Valid

Score
Clip

Category 1: Attack 1

Score
Clip

Category N: Attack N

Score
Clip

Largest Score at End Wins

OK

Not OK

Bloom Filters

Running Sum Score
Build Flow

- **Desire tools for automatic generation**
  - Infrequently rebuild and deploy
  - Utilize in other applications

- **User provides**
  - Labeled training data
  - Number of dictionary terms
  - Number of hash functions
  - Quantization levels
  - Bloom filter error rate

- **Tool chain generates header files**
  - C header or VHDL package
  - Static classifier software/hardware
  - Requires a rebuild of design
Performance Measurements

- Built and tested on Xilinx ML555 board
  - Xilinx Virtex5 LX50T -1 FPGA
  - Target GigE speeds (125MB/s)
  - Maximum clock rate: 196MHz

- Bottleneck: Input tokenizing/hashing
  - Byte stream interface
  - Append each token with 2-byte length during hashing
  - Results in extra stall cycle between tokens

<table>
<thead>
<tr>
<th>Situation</th>
<th>Efficiency</th>
<th>Rate @ 196MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst case</td>
<td>0.50</td>
<td>98 MB/s</td>
</tr>
<tr>
<td>Best case</td>
<td>0.99</td>
<td>194 MB/s</td>
</tr>
<tr>
<td>Average for actual data</td>
<td>0.85</td>
<td>166 MB/s</td>
</tr>
</tbody>
</table>
Resource Utilization

**Block RAM Utilization**

![Graph showing Block RAM Utilization with different quantization levels.](image)

**Slice Utilization**

![Graph showing Slice Utilization with different quantization levels.](image)

Slices as RAM
Future Directions

• **Architecture improvements**
  – Hybrid hashing: employ efficient hash structure for housing one-offs
  – Transition from compile-time data to run-time data

• **Additional platforms**
  – Tilera: Assign Bloom filters to different processor cores
  – GPU: Possible, but less attractive due to lack of network options

• **Application**
  – Apply to other data classification applications
  – Continued work in applying data classification techniques
Examining the Algorithm

• Term-Frequency, Inverse Document Frequency
  – TF: How often does each term appear in an attack?
  – IDF: How specific is the term to an attack?

\[
\text{tfidf}(t,d) = \frac{\sum \text{count}(t,d)}{\sum \text{count}(v,d)} \cdot \log \frac{|D|}{|\{d_j : t \in d_j\}|}
\]

• Cosine Similarity
  – Vector dot product: estimate angle between input and each attack category

\[
\text{sim}_{\text{cos}}(a,R) = \frac{\tilde{a} \cdot \tilde{R}}{\|\tilde{a}\| \|\tilde{R}\|} = \frac{\sum_{t \in a \cap R} \text{tfidf}(t,a) \cdot \text{tfidf}(t,R)}{\sqrt{\sum_{t \in a} \text{tfidf}(t,a)^2} \sqrt{\sum_{t \in R} \text{tfidf}(t,R)^2}}
\]