Towards Content-Centric Control Plane Supporting Efficient Anomaly Detection Functions

Hoang-Long Mai\textsuperscript{1,2} Guillaume Doyen\textsuperscript{2} Wissam Mallouli\textsuperscript{1}
Edgardo Montes-de-Oca\textsuperscript{1} Olivier Festor\textsuperscript{3}

\textsuperscript{1}Montimage, Paris - France,  \textsuperscript{2}Troyes University of Technology, France, \textsuperscript{3}Lorraine Research Laboratory in Computer Science and its Applications, France

Journées Cloud 2019, Toulouse, France
To appear in the 15th International Conference on Network and Service Management - CNSM 2019
Outline

1. **Introduction**

2. **Background**

3. **A content-centric Bayesian Inference Algorithm**

4. **Numerical Results**

5. **Conclusion**
Introduction

Context & Problematic
- Large number of low resource IoT devices & complex security functions
- Distributed Anomaly detection in IoT, Fog Computing

Hypothesis
- Each node executes identical security functions
- Metrics do not change frequently, results are repeated gradually over time
- Results are probably already executed in neighbor nodes
⇒ Hypothesis verification: *In a normal condition of Named Data Networking, 87% of computational security operations are repeated.*

Contribution
- Leveraging Named Function Networking (NFN) as an execution environment for anomaly detection (AN)
- Consider Bayesian Network (BN) inference as an AN framework since it stands for a representative function that numerous security components
Background on ICN - NDN - NFN

Information Centric Networking (ICN) - Named Data Networking (NDN)

- Name each content object instead of using IP address
- In-network caches for better delivery performance
- Among all ICN proposals, Named Data Networking (NDN) is the most promising one

Named function networking (NFN)

- Naming a function defined by its name and its parameters, and using \( \lambda \)-expressions as name resolution
- In-network caches results of calculation
Background on Bayesian Networks

**Terminology**

- A random variable, called $X_i$, is a set of possible values of a random phenomenon.
- An evidence $E = e$ is a subset $E = (X_{e_1}, \ldots, X_{e_m})$ of random variables standing for the observed phenomenon.
- A factor $\phi$ is defined as a function from a set of random variables $\text{Val}(X_1, \ldots, X_n)$ to $\mathbb{R}$.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>$\phi(A, B, C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.35</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0.08</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0.16</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0.15</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0.21</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.09</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0.18</td>
</tr>
</tbody>
</table>
Background on Bayesian Networks

Bayesian Inference

- A Bayesian Network (BN) is a probabilistic graphical model that consists of nodes and directed edges.

- The inference designates an algorithm which consists in calculating the conditional probability of $P(X_q|E=e)$:
  \[
P(X_q) = \frac{1}{Z} \sum_{X_n} \phi_n \cdot (\ldots (\sum_{X_2} \phi_3 \cdot (\sum_{X_1} \phi_2 \cdot \phi_1))) \tag{1}
  \]

  where
  \[
  Z = \sum_{X_1, \ldots, X_n} \prod_{i=1}^{n} \phi_i(X_i, Pa(X_i)) \tag{2}
  \]

Figure: Example of a Bayesian network
Background on Bayesian Networks

Variable Elimination (VE) algorithm

Input: initial factors (Φ) and evidence (E=e)
Output: Conditional probability \( P(X_q|E = e) \)

1. foreach \( \phi_i \in \Phi \) do
2. \( \phi_i \leftarrow \phi_i(E = e) \) // Factor reduction
3. end
4. Select Elimination Order (\( \sigma \));
5. foreach \( x_i \in \sigma \) do
6. \hspace{1em} foreach \( \phi_j \in \Phi \) do
7. \hspace{2em} if \( x_i \in \text{Scope}[\phi_j] \) then
8. \hspace{3em} \( \psi_i \leftarrow \psi_i \ast \phi_j \) // Factor product
9. \hspace{2em} end
10. \end
11. \( \phi_i \leftarrow \sum_{X_i} \psi_i \) // Factor marginalization
12. end
13. foreach \( \phi \in \Phi \) do
14. \( \varphi \leftarrow \varphi \ast \phi \) // Factor product
15. end
16. \( Z \leftarrow \sum_{X_1...X_n} \varphi \)
17. \( P \leftarrow \varphi/Z \) // Factor normalization

Algorithm 1: Variable Elimination algorithm
1 Introduction

2 Background

3 A content-centric Bayesian Inference Algorithm

4 Numerical Results

5 Conclusion
Naming scheme and data structure - Factor

### Properties
- The *factor* packet includes three parts: the list of the names of random variables, their dimensions, the list of all values of $\phi$.
- Types of factors: the initial factors and the temporary factors.

### Naming scheme
- `/data/fac/initial/<name of variables>`
- `/data/fac/temporary/<hash(factor)>`

### Data structure

<table>
<thead>
<tr>
<th>Structure</th>
<th>List of variables’ name</th>
<th>List of variables’ dimension</th>
<th>List of values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail</td>
<td>X1 X2 ... Xn dim(X1) dim(X2) ... dim(Xn)</td>
<td>$\phi(X_1 = 1, X_2 = 1, ..., X_n = 1)$ $\phi(X_1 = 1, X_2 = 1, ..., X_n = 2)$ ... $\phi(X_1 = \text{dim}(X1), X_2 = \text{dim}(X2), ..., X_n = \text{dim}(Xn))$</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>A B C 3 2 2</td>
<td>0.25 0.35 0.08 0.16 0.05 0.07 0.0 0.15 0.21 0.09 0.18</td>
<td></td>
</tr>
</tbody>
</table>

**Figure:** Data structure of a *factor*
Properties

- The packet of *evidence* encompasses three parts: the name of variables, their dimensions, and values of the *evidence*.
- An *evidence* is designed to be computed with a *factor* in *factor reduction*.
- The value of the *evidence* for a variable in a packet consists of a chain of 0’s and 1’s.

Naming scheme

- `/data/evi/<name of variables>/<values of evidence>`

Data structure

Figure: Data structure of an *evidence*
Transformation of functions into $\lambda$-calculus

**Principle functions**

- Factor reduction: `/func/reduce/(/data/fac/...,/data/evi/...)`
- Factor product: `/func/product/(/data/fac/...,/data/fac/...)`
- Factor marginalization: `/func/marginalize/(/data/fac/...,variable)`
- Factor normalization: `/func/normalize/(/data/fac/...)`

**Figure:** Example of function factor reduction

```plaintext
1  Function reduceFactor(fac, evi)
2      return getVarDim(fac,getVarDimEvi(evi)) + SEPARATOR_CHAR
3      serialize(filter(lambda x: x > -1,  
4          map(lambda x,y: -1 if y == 0 else x*y, 
5          deserialized(fac), reduce(lambda x,y:  
6           productEviVal(x,y),getValEvi(evi)))))
```

**Figure:** Transformation of factor reduction into $\lambda$-calculus
Outline

1. Introduction
2. Background
3. A content-centric Bayesian Inference Algorithm
4. Numerical Results
5. Conclusion
Use case: the Content Poisoning Attack

- Alter content by inserting Bad Data into the cache of routers
- CPA has a subtle impact on several of metrics

**Figure:** Experiment topology
Implementation

Figure: Bayesian Network to detect CPA
Implementation

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFN</td>
<td>The newest version of NFN written in python</td>
<td><a href="https://github.com/cn-uofbasel/PiCN">https://github.com/cn-uofbasel/PiCN</a></td>
</tr>
<tr>
<td>NFD</td>
<td>The Networking Forwarding Daemon of NDN</td>
<td><a href="http://named-data.net/doc/NFD/current/">http://named-data.net/doc/NFD/current/</a></td>
</tr>
<tr>
<td>jNDN</td>
<td>A NDN client library for Java</td>
<td><a href="https://github.com/named-data/jndn">https://github.com/named-data/jndn</a></td>
</tr>
<tr>
<td>pgmpy</td>
<td>Standard library for Bayesian Network</td>
<td><a href="https://github.com/pgmpy/pgmpy">https://github.com/pgmpy/pgmpy</a></td>
</tr>
</tbody>
</table>

Table: List of tools used in the implementation

Verification of proposed algorithm

The results of the proposed algorithm are 100% identical to the results of the standard library for BN - `pgmpy`. 
Evaluation

Figure: Snapshot of computational time over time
Evaluation

![Graph: Percentage of cache, local computation and communication]

**Figure:** Evolution of proportion of requests over time
Evaluation

Figure: Snapshot of computational time before and during attack
Evaluation

Figure: Impact of available CPU resources
Evaluation

Figure: Impact of \#nodes in BN
Evaluation

**Figure:** Impact of #relations in BN
Evaluation

Impacts on computational time

In term of computational time, the proposed approach is efficient when:

- The cache is fed with results from previous executions or neighbor nodes
- In normal traffic but not abnormal traffic
- The computation capacity is limited
- The BN is complex and needs a significant amount of operations
Outline

1. Introduction
2. Background
3. A content-centric Bayesian Inference Algorithm
4. Numerical Results
5. Conclusion
Conclusion & future work

Conclusion

- The core elements for the design and implementation of an NFN-supported Bayesian Network inference algorithm has been proposed.
- Numerical results demonstrated that an NFN-supported BN performs better not only in the case of limited computational resources but also when the BN is complex.

Future work

- Focus on further developing the content-oriented control plane by extending the current approach to integrate other methods for anomaly detection.
- Other attacks will also be considered to demonstrate the applicability and generality of the proposed approach.