IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 18, 2018, accepted April 14, 2018, date of publication April 25, 2018, date of current version May 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2829910

On Multilateral Security Monitoring and Analysis With an Abstract Tomogram of Network Flows

YOUNG YOON AND YONGJUN CHOI
Department of Computer Engineering, Hongik University, Seoul 04066, South Korea

Corresponding author: Young Yoon (young.yoon@hongik.ac.kr)

This research was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF), Ministry of Education, under Grant NRF-2016R1D1A1B03931324, in part by the 2018 Hongik University Research Fund, and in part by Netcore Tech Inc.

ABSTRACT In this paper, we present a novel method for visualizing an abstract tomogram of network flows. Through a tomogram, we offer visual cues for quickly sensing aggregate and temporal networking behaviors of the monitored systems. In an integrated view of a tomogram, users can cut across a specific dimension to reason about interesting networking activities without losing the overall picture of the networked system. We extend this tomogram with user interfaces for finding correlation between network flows and their attributes using a co-occurrence analysis algorithm. This paper also presents an interface for conducting sequence mining for interested flows in order to infer causal relationships. Security engineers need to prioritize the aforementioned situation analysis tasks by focusing on endpoints with relatively higher security risks. To help security engineers with this task, we devise a way to assess and visualize the security risks according to a new centrality measure that is computed based on various networking information from a set of network flows. Our paper shows that the novel visualization method and analytics interfaces offer more intuitive means to track down complicated symptoms of advanced and covert security threats.

INDEX TERMS Visual analytics, network tomography, tomogram, context awareness, network flows, co-occurrence, sequence mining, security risk, centrality.

I. INTRODUCTION

Security threats are constantly evolving to become more covert and complex. To detect the complicated symptoms of these advanced threats, it is necessary to analyze a myriad of heterogeneous security events [1]. However, field security engineers oftentimes have to go through the hassles of identifying the security events that require more urgent and thorough investigation. Visualizing the groups of security events and their basic statistical information may offer some hints about abnormal situations. However, existing visualization tools come short as they mostly offer fragmented and partial views of the complex security events. Also, these tools do not offer sufficient contextual information [2]. The having of Playstation network that occurred in 2011 revealed that security engineers who were pre-occupied with the analysis of many less-critical security events failed to recognize the actual moment of intrusion [3]. Such an unfortunate event calls for better visual cues and situation analysis interfaces. We have raised such concerns in our preliminary work that was presented in [4]

In this paper, we develop a way to provide an integrated view of security events that have occurred in a networked system. We refer to this comprehensive view as a *tomogram of network flows*, where security events are represented by network flows that are generated by a networked system. We adopt the term tomogram which is conventionally used as a representation of a cross-section through solid objects with penetration waves [5]. In our context, a tomogram of network flows (a tomogram for brevity) is a complete visual stack of various networking dimensions. For instance, a tomogram can show activities on pairwise connections between endpoints. At the same time, a tomogram shows aggregate and temporal networking behaviors in terms of protocols, ports and applications used by the endpoints, all within a single view. Given a tomogram, security engineers can cut across different networking dimensions without losing a macroscopic view of the entire system being monitored.

The main objective of this paper is to help the act of network tomography to be done in a more intuitive and contextual fashion. Here a network tomography is a study of a network's internal characteristics using networking information derived from endpoint data [6]. Given profiled behaviors in the past, our tomogram detects and highlights abnormal network flows that are subject to further investigation. In addition, our tomogram is equipped with user interfaces for analyzing situations around a flow of interest with co-occurrence and sequence mining algorithms. Our tomogram also visualizes criticality of an endpoint in terms of security risks. This paper provides a novel algorithm for computing multi-variate Eigenvector centrality of every node in the networked system based on the network flow information between endpoints. By highlighting highly critical nodes, security engineers can more easily prioritize additional tasks of analyzing situations around the flows that are related to the critical nodes.

This paper is structured as follows to elaborate our contributions further: (1) In Section 2, we introduce related works in the context of network tomography and visualization; (2) In Section 3, we explain a new tomogram visualization method; (3) In Section 4, we explain the tools that are implemented on the tomogram to help security engineers conduct situation analysis; (4) In Section 5, we present our new methods for assessing security risk of a node based on a centrality measure with network flow information; (5) In Section 6, we evaluate the performance of our methods, and (6) we conclude in Section 7.

II. RELATED WORK

In this section, we introduce related works on network tomography and security analysis for networked systems based on visualization of networking information.

Network tomography was first coined by Vardi [6] to define a study of network characteristics based on the information derived from endpoint data. Network tomography is mostly keen on performance studies such as link loss and packet latency that can be inferred from correlated end-to-end measurement [7]. Beside performance issues, network topologies can be discovered through tomographic inference [8]. This paper newly extends network tomography to the domain of network security. We represent the security events with network flows between endpoints. This paper devises a tomogram to visualize the network flows in multilateral fashion and to help security engineers in the field to conduct various statistical and situational analyses for sensing symptoms of advanced security threats.

Various visual tools exist for network management. Minarik and Dymacek [9] used a prober to extract statistics of Layer-3 communications between pairs of nodes in a network. Their tool shows information such as source, destination IPs and ports, packets and protocols used. The work by Liao *et al.* [10] showed a way to visualize the dynamics of a network. It used Gephi graph visualization tool [11] to observe any change in network conditions over time. It visualized bipartite host-to-host graphs and similarity graphs for analyzing the dynamics of transitions [10]. The works presented in [9] and [10] primarily rely on the visualization of pairwise communications and topological views. They do not support grouping of network nodes at various angles such as the use of applications, organizational information and protocols used. NVisionIP [12] is a tool that maps hosts on a two-dimensional (subnet $\times$ host) coordinate and clusters hosts by the characteristics of interest. NVisionIP can be used for recognizing suspicious port scanning activities and symptoms of denial-of-service attacks. However, with the abstraction of network conditions in a coordinate system, topological information gets lost. Therefore, it is difficult to monitor activities between endpoints. FlowTag provides query interfaces for filtering out interested event logs collected from security monitoring equipments [13]. Similar to [10], pairwise communications between two IP nodes can be analyzed. Query interface of FlowTag helps security engineers narrow down the logs to analyze. Hence analysis time could be significantly reduced. [14] visualized a treemap in a selected time span to show mostly used ports or hosts. Similarly, [15] showed a flow map where nodes are radially mapped around a focus node at the center. It used timelines and event plots for studying the temporal relationship between nodes. [16] divides a network into 4 layers according to subnet classes. Users can pinpoint a host and find any other hosts it interacted with. The abstraction of the network space is similar to the coordinate system of NVisionIP [12], and it is not easy for the users to spot a node of interest. Freet and Agrawal [17] visualized logs from IDS (Intrusion Detection System). However, the views are fragmented, which makes it difficult to see the collective behavior. Zhang *et al.* [19] used trees of HTTP requests and relationships on a radial map. However, this is limited to HTTP, and the aggregate behaviors on other network factors are not shown. Lastly, [19] presented inter-disciplinary research effort between security analysts, visual designers and data scientists on effectively coloring frequency of security log items on a multi-pixel structure. The visual cues are useful for easily recognizing abnormal events in the security logs. However, analyzing the context of such an event does not seem to be trivial.

From the previous works, we learned that excessive abstraction of the network condition may not provide intuitive visual analytics interface to the users. On the other hand, when the topological views are primarily used, overall statistical and contextual information may not be inferred easily. In this work, we aim to provide pairwise connection and contextual information at the same time and on the same view. We can show co-occurred network flows, frequently observed sequential patterns and correlations between network flow attributes. In addition, our tomogram can visualize significant changes at various networking perspectives during temporal transitions, without losing the overall picture of the whole system. In the following, we show how such novel visual analytics can be realized.

III. CONSTRUCTION OF NETWORK TOMOGRAM

In this section, we introduce a way to represent a set of network flows with a tomogram. A network flow is a record

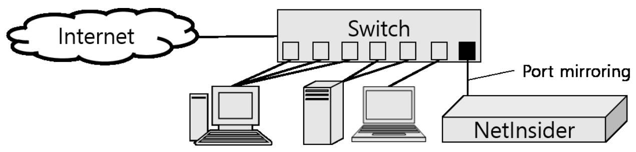


FIGURE 1. Collecting network flows by conducting deep packet inspection on all packets captured on a mirrored port.

of communication information between two endpoints [20]. In order to collect the network flows, we first use port-mirroring to capture all packets flowing through a network, as shown in Fig. 1. We feed the captured packets into an appliance that can conduct Deep Packet Inspection (DPI) [21]. We employed an enterprise-grade DPI machine (NetInsider) that can retrieve up to 40 million concurrent flows from a 40 Gbps network.[1] Besides standard Layer-3 information such as IPs and ports of communicating endpoints, we extract traffic volume, protocols and application types such as Web, Peer-to-Peer application, E-mail and HTTP used by the endpoints.

Algorithm 1 Generate Network Tomogram.

```
   Input: List of Flows, FlowList
   Output: Network tomogram
1  Function updatetomogram(Node, Flow):
2      if Attribute of Node is "eport" then
3          Update Node's flowCount, size, packetCount;
           return;
4      else if Flow is not in child nodes of Node then
5          Create a node n with Flow's flowCount, size,
           packetCount ;
6          Add n to the child nodes of Node;
7          Update Node's flowCount, size, packetCount;
8          updatetomogram(n, Flow);
9      else
10         Update Node's flowCount, size, packetCount;
11         Find a Node c that is in the child nodes of Node
           and contains Flow;
12         updatetomogram(c, Flow);
13 Function generatetomogram(FlowList):
14     Previous network tomogram, $T_p$;
15     A new empty network tomogram, $T_c$;
16     Copy $T_p$ to $T_c$;
17     foreach Flow in FlowList do
18         updatetomogram($T_c.Root$, Flow);
```

We collect these network flows periodically and generate a tomogram according to Algorithm 1. A flow is inserted into a tree that is dissected with different horizontal layers. Here, each layer represents an attribute of a network flow. *Day* is the time when the flow was retrieved. *norg* and *eorg* are destination and source organizations (or geographical

[1]Full specification available at http://www.netcoretech.co.kr

Flow ID	1	2	3	4	5
day	2017. 12.20	2017. 12.20	2017. 12.20	2017. 12.20	2017. 12.20
norg	2	2	2	2	2
eorg	999	999	999	999	999
app type	0	0	0	2	2
proto-col	6	6	6	6	6
eip	192.168 .0.2	192.168 .0.2	192.168 .0.2	192.168 .0.4	192.168 .0.4
nport	80	80	22	22	23
eip	2.2.3.1	2.2.3.2	2.2.3.3	2.2.3.2	2.2.3.4
eport	123	124	22	124	23

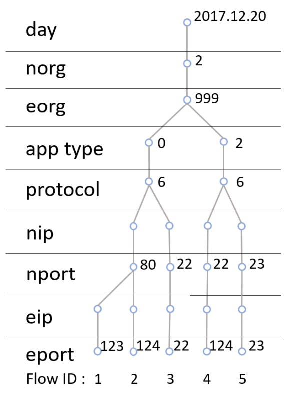


FIGURE 2. An example of tomogram construction.

Sequence of Flow ID :[1 3 4 2]5 3 2 2 ⋯ T_1 = {1, 3, 4, 2}

Sequence of Flow ID : 1[3 4 2 5]3 2 2 ⋯ T_2 = {3, 4, 2, 5}

Sequence of Flow ID : 1 3[4 2 5 3]2 2 ⋯ T_3 = {4, 2, 5, 3}

Sequence of Flow ID : 1 3 4[2 5 3 2]2 ⋯ T_4 = {2, 5, 3} , T_4 = {2, 5, 3}

⋮

FIGURE 3. An example of generating transactions from a sequence of flow IDs for co-occurrence analysis.

locations), respectively. The letters *n* and *e* stands for ingress and egress, respectively. We apply this notation for IP and port attributes as well for brevity, i.e., *nip, eip, nport* and *eport*. We use pre-defined code number for *app type* and *protocol* attributes.

An example of constructing a tomogram from a set of five network flows is given in Fig. 2. Suppose a new flow with ID $= 1$ and *app type* $= 0$ is inserted for the first time. There is no node with *app type* $= 0$ on the *app type* layer. In this case, our algorithm creates a new node on the *app type* layer with *app type* $= 0$. When a following flow with ID $= 2$ and *app type* $= 0$ is inserted, our algorithm spots a matching node at *app type* layer with *app type* $= 0$. Our algorithm traverses further down in the child layers to update the nodes that match the current flow being inserted until a leaf node at the *eport* layer is reached. While traversing through the tomogram, our algorithm updates flow count, flow size (in bytes) and packet count on the matching nodes. For instance, flow count of the node with *protocol* $= 6$ on the *protocol* layer is 3. Each branch in this tree structure represents a network flow.

We generate a tomogram periodically and compute a cumulative moving average (CMA) of flow count, flow size and packet count of every node for a given sequence of tomographies. We color a node with the CMA values as follows. If any CMA value on a tomogram node exceeds a configurable threshold at a certain period, then we color the node red. A red node indicates a sudden increase of networking activities. Suppose a node with *nport* $= 80$ is colored red, then we can infer that there is a sudden influx of flows on port 80. On the other hand, if any CMA value on a tomogram node falls below a configurable threshold, then we color the node grey to show it that it is ephemeral. For example, port 80 stopped to exhibit any networking activity, then the node with *nport* = 80 is colored grey. If an ephemeral node reappeared after some period, then the node is colored yellow to reflect a sporadic networking behavior. A newly created node is colored blue to indicate that a never-seen flow has appeared. If there was little change to a CMA value, then a node is colored white to represent a stationary behavior.

Any node that is not colored white is a sign of abnormal situation. Instead of looking into a myriad of individual flow events, we can focus on a specific layer and drill down to the child layers for more lower-level details. For instance, we can cut through the *app type* layer to pinpoint a problematic node. Then, we can navigate through the IP layers to list the endpoints involved in the problematic application and check whether they exhibit suspicious behaviors. Suppose the *app type* layer shows that it is the web application that behaves abnormally. Subsequently, we can move down to the IP layers to find out which web server is communicating excessively with external endpoints. Such symptoms can be a sign of potential data leakage. As opposed to many existing tools that rely on the visualization of the topology of endpoints [2], our tomogram intuitively visualizes aggregate and temporal behavior of network flows at different dimensions in addition to pairwise connection information, all in an integrated view.

IV. SITUATION ANALYSIS WITH NETWORK TOMOGRAM

In this section, we extend the tomogram to support analysis of contextual information surrounding network flows of interest. This section focuses on co-occurrence and flow sequence analysis.

A. CO-OCCURRENCE ANALYSIS

Given a set of network flows, we can list flows that co-occurred with a flow of interest. As shown in the example illustrated in Fig. 4, a user can select an interested flow by selecting its leaf node. After selection, we compute conditional probabilities of flows co-occurring with the selected flow. We highlight co-occurred flows on the tomogram and return the list of actual conditional probabilities. Such correlation can be useful for detecting an anomaly. For instance, we can compute the average distribution of co-occurrence for a specific flow in the past as shown in Table 1. The distribution in the past can be compared with the one obtained from a new time period. If there is a significant discrepancy, we can suspect an anomaly and trigger further investigation.

We compute the co-occurrence with FP-Growth algorithm [22]. In order to employ this algorithm, we need to design a transaction database (DB) as follows. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of flow IDs and a transaction $DB = \langle T_1, T_2 \ldots, T_i \rangle$, where $T_i (i \in [1..n])$ is a transaction that contains a set of flow IDs $\in I$.

Fig. 3 illustrates how transactions are extracted. We slide fixed-length time window over a sequence of flow IDs.

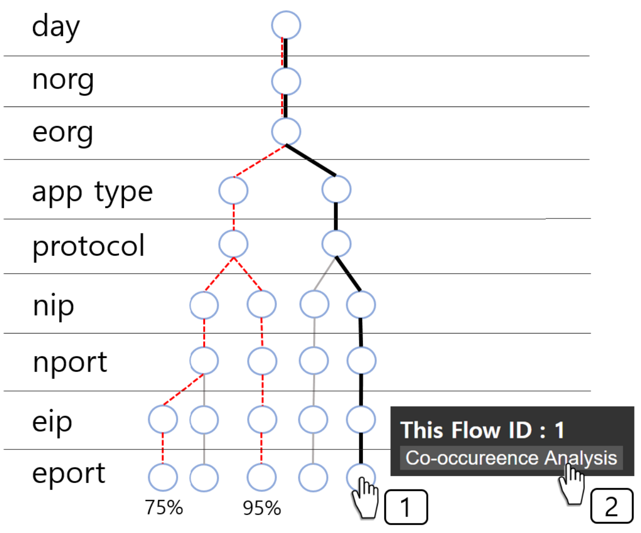


FIGURE 4. An example of highlighting flows that co-occurred with a selected flow of interest. The conditional probability of co-occurrence is also shown.

TABLE 1. Changes of co-occurrence distribution over time.

Past Average		Current	
FlowID	FlowID's Conditional probablility	FlowID	FlowID's Conditional probablility
4	1 : 50% 2 : 75% 3 : 60%	4	1 : 80% 2 : 80% 3 : 60%
6	5 : 60% 8 : 70% 11 : 90%	6	5 : 65% 8 : 65% 11 : 90%
7	9 : 80% 10 : 40%	7	9 : 80% 10 : 70%

All flow IDs within the time window is put into a transaction. If there are duplicate flow IDs, we multiply the transaction. For example, items in a time window [2 5 3 2] yields two duplicate transactions, $T_4 = \{2, 5, 3\}, \{2, 5, 3\}$. Such multiplication is to make sure we correctly capture transactions that are likely to co-occur more. We move the time window every time tick. Therefore, time windows can overlap each other.

We define a *support* value of a flow item *i* in a transaction as $\frac{freq\_i}{|T|}$ where *freq_i* is the frequency of *i*, and $|T|$ is the total number of transactions obtained from the set of network flows. Confidence is the conditional probability of an item co-occurring with another flow. We can set min_suppport and min_confidence. During the iterative computation of co-occurrence, we rule out items whose support or confidence value is less than min_support or lower than min_confidence. Less frequently appearing flows can be overlooked when min_support is set to a high value. Despite the low frequency, some of them may still pose security threats. However, we stress again that we color ephemeral, sporadic and never-seen flows to alert the security engineers and prevent less frequent but critical flows passing unnoticed.

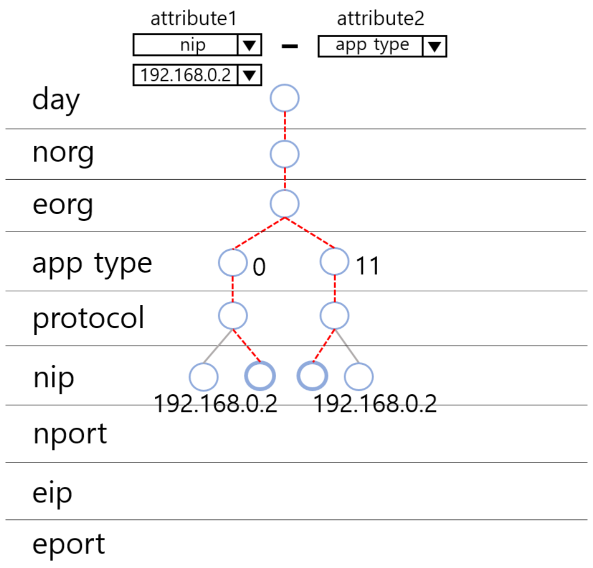


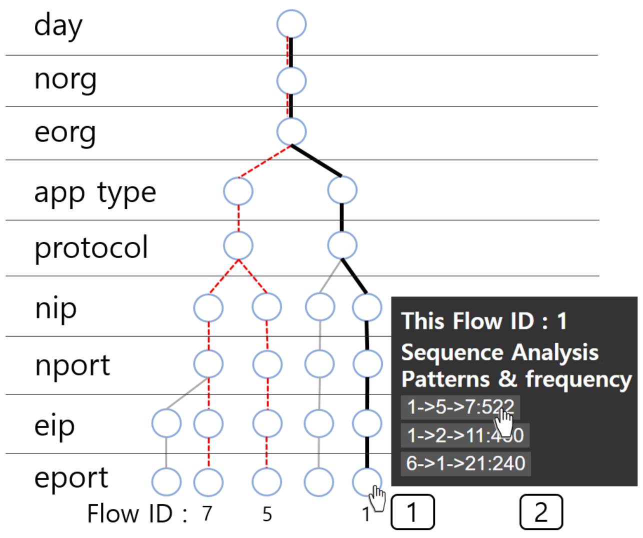
FIGURE 5. Showing correlated flow attribute values on a tomogram.

TABLE 2. Patterns of co-occurring attribute values over time.

Past Average		Current	
nip	*app type* Conditional probability	*nip*	*app type* Conditional probability
192.168.0.2	Web : 50% P2P : 50%	192.168.0.2	Web : 90% P2P : 10%
192.168.0.4	E-mail : 70% P2P : 30%	192.168.0.4	E-mail : 75% P2P : 25%
192.168.0.6	P2P : 80% Web : 20%	192.168.0.6	P2P : 80% Web : 15% E-mail : 5%

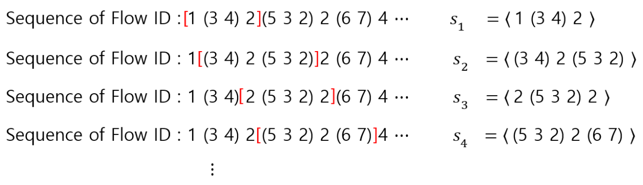
We can also analyze co-occurrence between different attributes. To do this, we provide another definition of transaction database. Let $I_a = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_\mathrm{m}\}$ be a set of values of a flow attribute $a \in \{norg, eorg, application, protocol, nip, nport, eip, eport\}$. We define transaction $DB = \langle T_1, T_2 \ldots, T_\mathrm{i} \rangle$, where T_i ($i \in [1..\mathrm{n}]$) is a transaction which contains two different attribute values that we want to correlate.

As shown in Fig. 5, users can select two attributes to correlate, e.g., *nip* and *app type*. Upon selection of *nip* = 192.168.0.2, all flows containing this value is retrieved. We run the FP-Growth algorithm in order analyze what *app type* co-occurred frequently with *nip* = 192.168.0.2. Fig. 5 shows the flows that went through *nip* = 192.168.0.2 and indicates that they were frequently involved in *app type* = 0 and 11. On the tomogram, we show only the correlated flows. Table 2 shows the past and current *app types* that co-occurred with selected *nip* values. Any significant difference can be suspected as anomaly. For instance, we can notice a 5% conditional probability of using E-mail on *nip* = 192.168.0.6 in the current period while no such activity was observed in the past. Such a change can be suspected as an anomaly.



This Flow ID : 1
Sequence Analysis
Patterns & frequency
1->5->7:522
1->2->11:400
6->1->21:240

FIGURE 6. Highlighting frequently appearing sequences of flows on a tomogram.



Sequence of Flow ID :[1 (3 4) 2](5 3 2) 2 (6 7) 4 ···	s_1 = ⟨ 1 (3 4) 2 ⟩
Sequence of Flow ID : 1[(3 4) 2 (5 3 2)]2 (6 7) 4 ···	s_2 = ⟨ (3 4) 2 (5 3 2) ⟩
Sequence of Flow ID : 1 (3 4)[2 (5 3 2) 2](6 7) 4 ···	s_3 = ⟨ 2 (5 3 2) 2 ⟩
Sequence of Flow ID : 1 (3 4) 2[(5 3 2) 2 (6 7)]4 ···	s_4 = ⟨ (5 3 2) 2 (6 7) ⟩
⋮	

FIGURE 7. A example of extracting sequence of elements from a set of network flows.

B. FLOW SEQUENCE ANALYSIS

In this section, we provide an interface for analyzing frequent sequences that occurred around a flow of interest. For instance, as shown in Fig. 6, a user can first select the leaf node of an interested flow (ID = 1). Frequently appeared sequences that include the selected flow are shown in the pop-up box and are also highlighted directly on the tomogram.

We employ Prefix-Span [23] algorithm for computing the frequently appearing sequences from a set of network flows. To use this algorithm, we defined an item set I and a sequence S as follows. Let $I = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_\mathrm{m}\}$ be a set of flow IDs. A sequence S is an ordered list of *element*s, which is denoted by $\langle s_1, s_2 \ldots, s_\mathrm{i} \rangle$. s_i is a subset of network flow IDs. An element contains one or more flow IDs that occurred at the same time. Fig. 7 illustrates an example of retrieving sequences of elements from a set of network flows.

We slide the fixed-length time window over a sequence of network flows at every time tick. Items in ( ) indicate flows that simultaneously appeared in the networked system. We can set the maximum length of the sequence pattern to observe. Table 3 shows more examples of sequences that include interested flow IDs. With this information, security engineers can infer causal relationship among flows.

Co-occurrence and sequence mining results obtained through our tomogram can be considered as a set of features that represent the situation of a monitored networked

TABLE 3. Sequence Analysis Example.

Flow IDs of interest	Related Patterns	Frequency
1	1 -> 5 -> 7 1 -> 2 -> 11 6 -> 1 -> 21	522 460 240
3	3 -> 9 -> 13 3 -> 7	550 440
7	7 -> 24 -> 18 3 -> 7 7 -> 2 -> 8	520 440 220

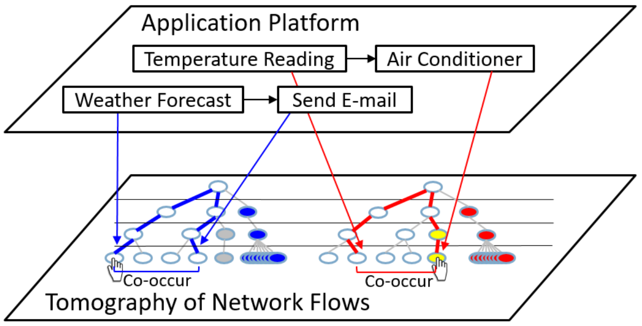


FIGURE 8. Mapping between application workflow and a sequence of network flows on the tomogram.

system at any given time. As a future work, we can extend our tomogram to support tracking and modeling the temporal transition of the features using regressions, distributed Bayesian network algorithms or LSTM networks [24]–[26] in order to sense anomalies more effectively. In addition, we can add another visual layer on top of the tomogram that shows application logic. In [27], we devised a way to correlate the sequence of network flows and description of interactions between Web of Things (WoT) services. As illustrated in Fig. 8, we can show the mapping of application workflows on the tomogram. If there is a flow sequence that does not match a registered workflow of service interaction at the application layer, then we can immediately alarm this as an anomaly.

V. TOMOGRAM WITH NODE CENTRALITY

This section presents a visual cue to help security engineers select more critical flows for further investigation with higher priority. A criticality of a flow can be measured in terms of security risks on an endpoint that is involved in the flow. SP 800-30 by NIST provides a framework for assessing risks of information technology systems [28]. The downside of this framework is that the risk assessment is done manually by humans, and it can, therefore, yield subjective results. Also, the framework does not sufficiently account for the various networking activities that can contribute to the risk.

This article devises a more systematic approach to ranking the security risks of an endpoint by a new centrality measure that is based on endpoint's various networking behaviors.

First, some terms are defined as follows. *NF* is a set of network factors such as flow volume, the number of flows,

Algorithm 2 Node Ranking Algorithm

```
Input: Networked nodes $G$, Max Iteration $iteration$
Output: Scores of all nodes in $G$
1  Function NodeRank ($G$, $iteration$):
2      $d \leftarrow 0.85$;
3      Network Factors $NF$;
4      foreach factor $i$ in $NF$ do
5          $F_i \leftarrow \sum$ (values of $i$ outbound flows);
6          foreach node $v$ in $G$ do
7              Initialize score $R_f(v)$ to $\frac{1}{OB}$;
8      foreach factor $i$ in $NF$ do
9          while $iteration > 0$ or not converged do
10             $F_i \leftarrow \sum$ (values of $i$ on outbound flows);
11             foreach node $v$ in $G$ without any flows do
12                 $R_f(v) \leftarrow R_f(v) + d * \frac{1}{F_i}$;
13             foreach node $v$ in $G$ do
14                 foreach neighbor $n$ in neighbors($v$) do
15                     $F_n \leftarrow \sum$(outbound flows from $n$ to $v$);
16                     $R_f(v) \leftarrow R_f(v) + \frac{1-d}{F_i} + d * \sum \frac{R_f(n)}{F_n}$;
17                     $R_f(n) \leftarrow R_f(n) + \frac{1-d}{F_i} + d * \sum \frac{R_f(v)}{F_n}$;
18             iteration = iteration - 1;
19     foreach factor $f$ in $NF$ do
20         foreach node $v$ in $G$ do
21             Total Rank of $v$, $TR(v) \leftarrow 0$;
22             $W_f \leftarrow$ weight of $f$;
23             $TR(v) = TR(v) + W_f * R_f(v)$;
```

the number of applications running on a node, the number of open ports and communication protocols. A centrality value of a node is a weighted sum as defined in Equation 1.

$$R(v_x) = \sum_{i \in NF} \alpha_i R_i(v_x) \tag{1}$$

A weight α_i of network factor i is set to 1 by default. Note that α_i can be flexibly configured by the security engineers who may want to impose different weight values to network factors in certain application domains. Determining optimal $alpha_i$ values requires another research effort of assessing the effects of different weight combinations to security analysis in various domains. This is not in the scope of this paper, and we leave it out for a future research when enough cases are collected.

$R(v_x)$ is score of v_x, which is defined in Equation 2.

$$R_i(v_x) = \frac{1-d}{n} + d\Big(\sum_{t \in G} a_{i(v,t)} \cdot \frac{R_i(v_t)}{C_i(v_t)}\Big) \quad (i \in NF) \tag{2}$$

$a_{i(v,t)}$ is an adjacency matrix as well as the stochastic matrix whose sum of every row is 1. $C(v_t)$ is for normalizing the rank value, which is defined in Equation 3.

$$C_i(v_t) = \Big(\sum_{t=1}^{n} a_{i(v,t)}\Big)(i \in NF) \tag{3}$$

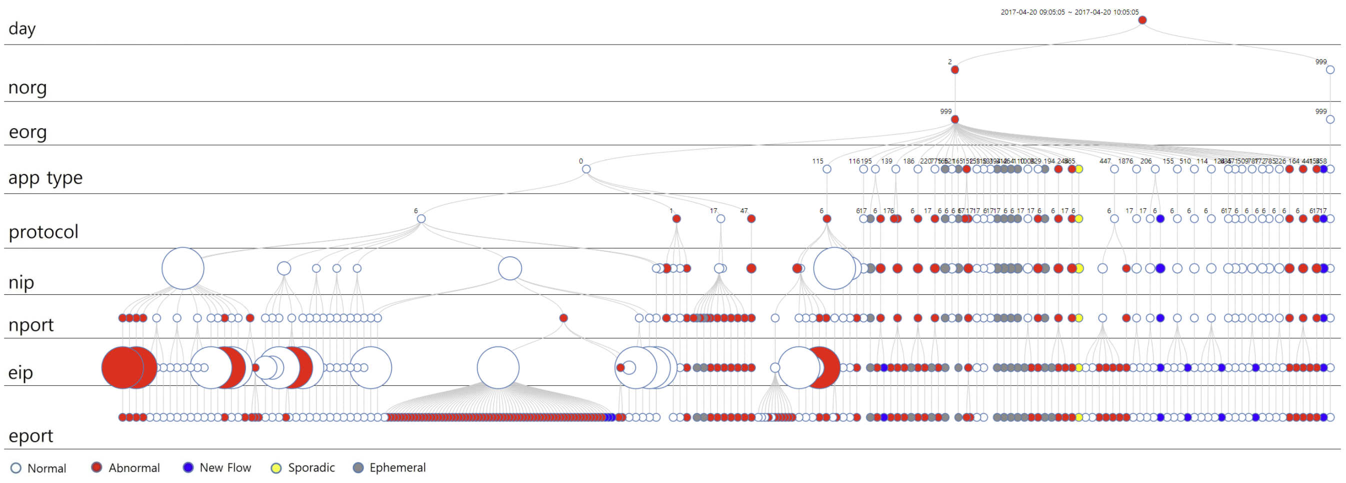


FIGURE 9. A sample tomogram of network flows implemented with D3.js. Size of a node and the layers for endpoints indicate security risks in terms of centrality.

Our newly defined centrality can be regarded as a multivariate version of Eigenvector centrality [29] and an adaptation of PageRank [30]. Note that d in Equation 2 is a constant called a damping factor. Damping factor reflects the phenomenon that not all endpoints are reached directly from other endpoints. We set the damping factor to 0.85, which is the value used by PageRank as well to discount scores of incoming page links by 15%. While PageRank considers only the incoming links for scoring the rank of a node, we consider both the outbound and inbound flows for assessing the node's centrality. Outbound flows from a node can potentially indicate a data leakage, while inbound flows can pose potential threats such as intrusions and denial-of-service attacks. The pseudo-code for our centrality computation algorithm is given in Algorithm 2.

On our tomogram, the centrality of an endpoint is expressed as the size of a circular node. An example of a tomogram with centrality information is shown in Fig. 9. This is implemented with D3.js which is a Javascript library for visualization of Big Data [31]. As we can see in the figure, we can easily spot critical nodes with high centrality (high risk). Any flows flowing through the large non-white nodes can be selected with high priority for further situation analysis.

VI. EVALUATION

In this section, we measure the overhead of conducting the situation analyses and computing the node centrality. All of our analysis and computations were executed on a 64-bit CentOS Linux release 7.3.1611 (Core) machine with four Intel i5 CPUs @ 3.20GHz and 4GB of RAM. We wrote the tomogram generation program in Java. A web-based interactive visualization of the tomogram was implemented in Javascript with D3.js library [31]. We used Spark Machine Learning Library (MLLib) [32] to implement FP-Growth for co-occurrence analysis and PrefixSpan for sequence mining. We employed the NetInsider DPI appliance in order to extract network flows from randomly generated network packets. We indexed network flows in ElaticSearch, a distributed document database [33].

The sample target system we observed with our tomogram was a Big Data platform for real-time analytics which is deployed in our lab network, as shown in Fig. 16. This platform uses Kafka [34] message queue for a real-time data ingestion. Data collected by Kafka is further relayed to Spark [35] applications that are scheduled to run parallel tasks on a cluster of computing resources managed by Mesos [36]. Mesos master is considered as the most critical component in this case, as its failure can cause unavailability of the entire Spark applications. The arrows in Fig. 16 represent a direction of flow between two endpoints. The number on each arrow head is the number of the total flow between two endpoints collected for an hour. The number of total flow is one of the network factors used to compute the node centrality with a program that was fully written in C++.

The latency of generating tomogram with varying number of flows is shown in Fig. 10.

It took approximately 183 seconds to construct a tomogram with 724,000 flows, which is 17 hours worth of flows appeared in our networked system. All child nodes in a tomogram layer are hashed for a fast search of matching flows while constructing and updating a tomogram. This helped us maintain linear latency growth to the number of flows in the system.

In Fig. 11, we measured the latency of computing flow co-occurrences, correlating between flow attributes and analyzing sequences of flows (denoted FPG-Flow, FPG-Attribute, and PrefixSpan-Flow, respectively). We set min_support and min_confidence to 0.8 for both FP-Growth and PrefixSpan algorithms. We set the maximum length of the sequence pattern (MaxPatternLength) to 5 for PrefixSpan algorithm. With FPG-Attribute, every transaction contains two flows.

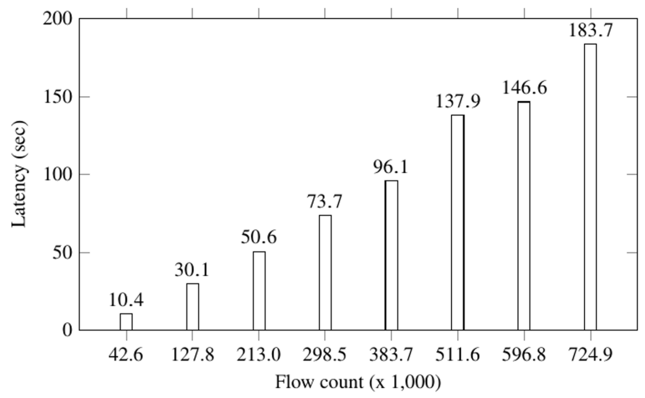


FIGURE 10. Latency of tomogram generation.

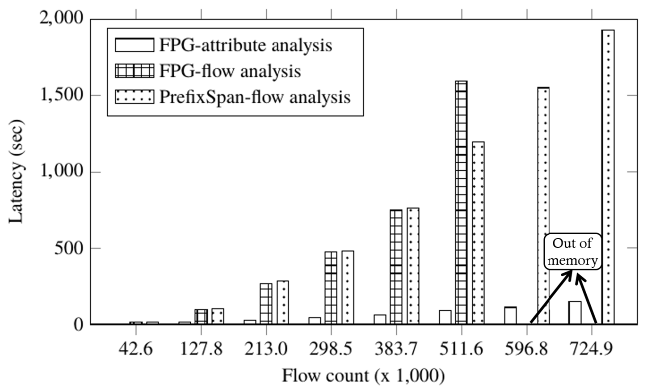


FIGURE 11. Latency of situation analyses with varying number of flows.

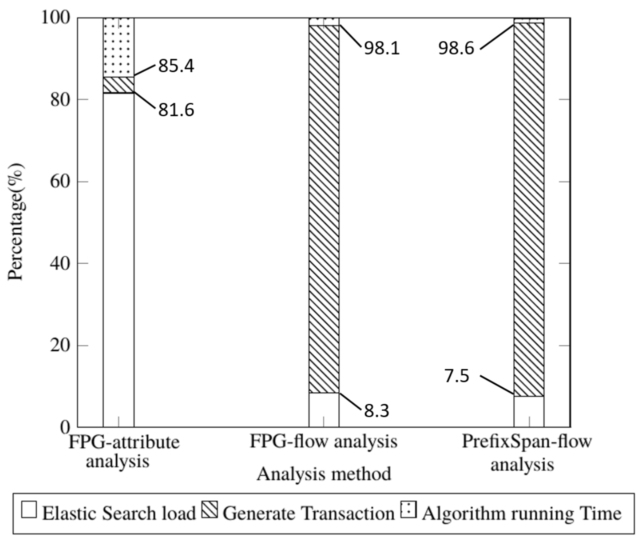


FIGURE 12. Latency profiling for situation analysis methods.

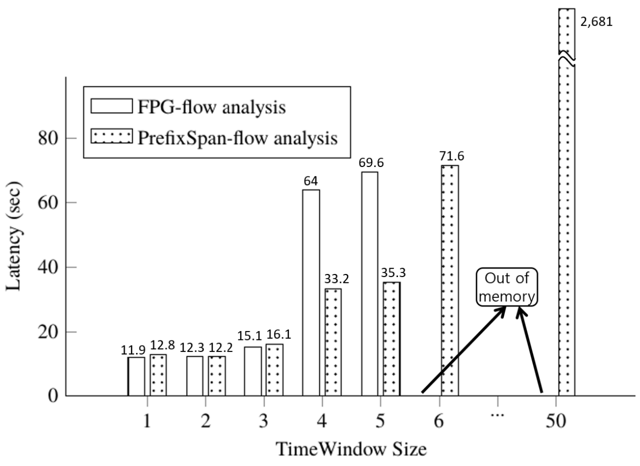


FIGURE 13. The effect of time window on latency of situation analyses.

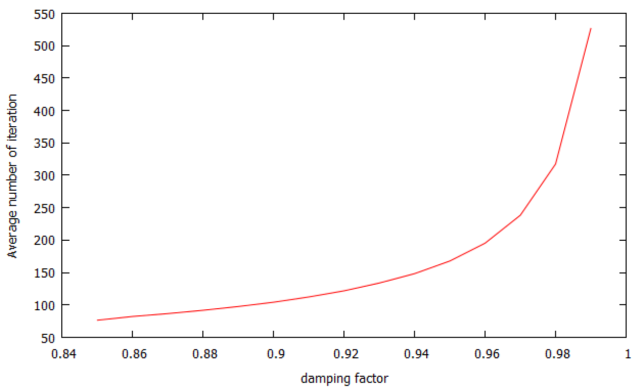


FIGURE 14. Number of iterations with varying damping factor.

For other analysis methods, we fixed the time window size to 3 for retrieving transactions.

As shown in Fig. 11, FPG-Attribute analysis took relatively less time as only the subset of flows that match the interested attribute-value were retrieved. The latency increased linearly with the increase of flows for the other two analysis methods. However, when there were more than 596,800 flows to analyze, FPG-Flow suffered out-of-memory errors.

As shown in Fig. 12, FPG-Flow analysis and PrefixSpan-Flow analysis spent the most time in generating transactions. For both methods, the proportion spent on running the actual algorithm was relatively small, while I/O cost of retrieving flow data from the flow storage (ElasticSearch) was dominant for FPG-attribute analysis.

As shown in Fig. 13, we measured the analysis latency with varying time window size in seconds for FPG-Flow and PrefixSpan-Flow. In this measurement, the total number of flows to analyze was fixed at 42,000. FPG-Flow failed due to lack of memory when the time window size was increased beyond 6. On the other hand, PrefixSpan-Flow suffered out-of-memory error when the time window size was increased to 50. As time window size increases, there is a higher chance of the same flow repeatedly showing up within the time window. Therefore, the number of transactions to generate grew significantly as well.

As for the performance of our node centrality computation, we first measured the number of iterations against damping factors from 0.85 to 0.99. As shown in Fig. 14, the number

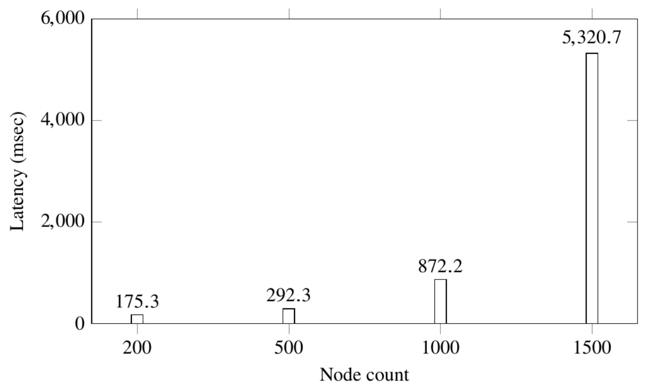


FIGURE 15. Latency of node centrality computation.

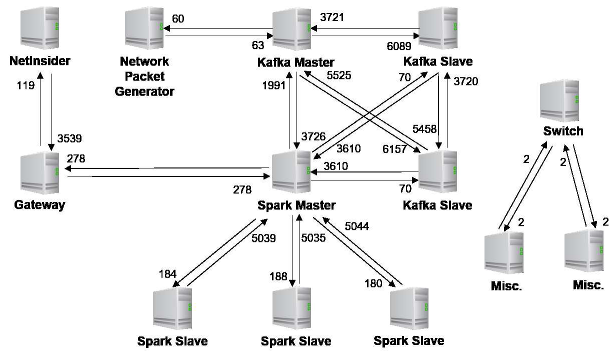


FIGURE 16. Network topology and network flow information of our internal Big Data processing platform.

of iterations spiked at 0.99. This observation shows that damping factor of 0.85 is reasonable in terms of the number of iterations.

In Fig. 15, we show the latency of computing node centrality. We observed sudden spike when the number of nodes increased to 1,500. The high overhead for computing the centrality is inevitable especially when there are multiple networking factors to consider. As a future work, we plan to benchmark existing parallelization techniques such as [37] in order to devise a new parallel algorithm for our centrality computation.

Lastly, in Table 4, we verified whether our centrality algorithm correctly measured the importance of every component in our Big Data platform. We ranked each node by its centrality and found out that Mesos master and the Gateway nodes ranked the highest. As mentioned earlier, failure of Mesos master can cause a disruption to entire Spark applications. The Gateway node is the first entry point to the systems in the internal private network. Infiltration into the Gateway could lead to cascading intrusions to all other internal nodes. We can see that the rankings reasonably reflected the importance of key components of our Big Data processing platform.

Note that the nodes in our internal network are ranked relatively higher than the nodes on the external network (e.g., Internet). This is because we only port-mirrored our own networking switch. Unless we tap into the core switches of the entire Internet, we cannot observe the networking activities among all external nodes. We can extend our tomogram to highlight endpoints from external networks and rank their centrality separately so that we can monitor external threats more effectively.

TABLE 4. Rankings of each node in our Big Data processing platform by our new centrality values.

IP address	Role	Centrality	rank
192.168.0.2	Gateway	0.12974463	2
192.168.0.4	Spark Master	0.151899096	1
192.168.0.5	Spark Slave	0.015398213	9
192.168.0.7	Spark Slave	0.015913387	8
192.168.0.8	Spark Slave	0.015375129	10
192.168.0.10	NetInsider	0.049231309	6
192.168.0.12	Packet generator	0.017043904	7
192.168.0.18	Misc. PC	0.010282737	13
192.168.0.19	Misc. PC	0.010291066	12
192.168.0.21	Kafka Master	0.128421207	3
192.168.0.22	Kafka Slave	0.105954794	4
192.168.0.23	Kafka Slave	0.088479635	5

VII. CONCLUSION

We devised a novel method for collectively visualizing aggregate and temporal behaviors of a network system from a set of network flows that were captured with a DPI appliance. We call this visualization a tomogram, and we extended it to help security engineers conduct additional situation analysis. Users can find flows that co-occurred with a flow of interest, find a correlation between network attribute values and analyze a frequent sequence of flows to infer causal relationship among flows. We can also help the security engineers prioritize the flow analysis tasks by highlighting the endpoints with a higher risk on the tomogram. We assessed the security risk of a node in terms of a multi-variate centrality measure we newly defined with various networking behaviors.

As a future work, we plan to extend our tomogram to model network situations with various state-of-the-art machine learning algorithms.

REFERENCES

[1] B. E. Ulicny, J. J. Moskal, M. M. Kokar, K. Abe, and J. K. Smith, "Inference and ontologies," in *Cyber Defense and Situational Awareness*. Cham, Switzerland: Springer, 2014, pp. 167–199.

[2] Y. Zhonghua and W. Lingda, "Summary on network security visual analysis," in *Proc. 7th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Aug. 2016, pp. 989–991.

[3] K. Poulsen, "Playstation network hack: Who did it," *Wired New*, 2011. [Online]. Available: https://www.wired.com/2011/04/playstation-hack/

[4] Y. Yoon, Y. Choi, and S. Shin, "Multilateral context analysis based on the novel visualization of network tomography: Poster," in *Proc. 11th ACM Int. Conf. Distrib. Event-Based Syst.*, 2017, pp. 343–344.

[5] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*. Cham, Switzerland: Springer, 2009.

[6] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *J. Amer. Statist. Assoc.*, vol. 91, pp. 365–377, 1996.

[7] N. Duffield, "Network tomography of binary network performance characteristics," *IEEE Trans. Inf. Theory*, vol. 52, no. 12, pp. 5373–5388, Dec. 2006.

[8] B. Eriksson, G. Dasarathy, P. Barford, and R. Nowak, "Efficient network tomography for Internet topology discovery," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 931–943, Jun. 2012.

[9] P. Minarik and T. Dymacek, "NetFlow data visualization based on graphs," in *Visualization for Computer Security*. Berline, Germany: Springer, 2008, pp. 144–151.

[10] Q. Liao, A. Striegel, and N. Chawla, "Visualizing graph dynamics and similarity for enterprise network security and management," in *Proc. ACM 7th Int. Symp. Vis. Cyber Secur. (VizSec)*, New York, NY, USA, 2010, pp. 34–45. [Online]. Available: http://doi.acm.org/10.1145/1850795.1850799

[11] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Proc. ICWSM*, vol. 8. 2009, pp. 361–362.

[12] K. Lakkaraju, W. Yurcik, and A. J. Lee, "NVisionIP: NetFlow visualizations of system state for security situational awareness," in *Proc. ACM Workshop Vis. Data Mining Comput. Secur.*, 2004, pp. 65–72.

[13] C. P. Lee and J. A. Copeland, "Flowtag: A collaborative attack-analysis, reporting, and sharing tool for security researchers," in *Proc. ACM 3rd Int. Workshop Vis. Comput. Secur.*, 2006, pp. 103–108.

[14] F. Fischer and D. Keim, "VACS: Visual analytics suite for cyber security-visual exploration of cyber security datasets," in *Proc. IEEE VIS*, Oct. 2013, pp. 13–18.

[15] D. Phan, J. Gerth, M. Lee, A. Paepcke, and T. Winograd, "Visual analysis of network flow data with timelines and event plots," in *Proc. VizSEC*, 2007, pp. 85–99.

[16] T. Itoh, H. Takakura, A. Sawada, and K. Koyamada, "Hierarchical visualization of network intrusion detection data," *IEEE Comput. Graph. Appl.*, vol. 26, no. 2, pp. 40–47, Mar. 2006.

[17] D. Freet and R. Agrawal, "A virtual machine platform and methodology for network data analysis with IDS and security visualization," in *Proc. IEEE SoutheastCon*, Mar./Apr. 2017, pp. 1–8.

[18] H. Zhang, M. Sun, D. D. Yao, and C. North, "Visualizing traffic causality for analyzing network anomalies," in *Proc. ACM Int. Workshop Int. Workshop Secur. Privacy Analytics*, 2015, pp. 37–42.

[19] J. Landstorfer, I. Herrmann, J.-E. Stange, M. Dörk, and R. Wettach, "Weaving a carpet from log entries: A network security visualization built with co-creation," in *Proc. IEEE Conf. Vis. Analytics Sci. Technol. (VAST)*, Oct. 2014, pp. 73–82.

[20] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a better NetFlow," in *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 245–256, 2004.

[21] P.-C. Lin, Y.-D. Lin, Y.-C. Lai, and T.-H. Lee, "Using string matching for deep packet inspection," *Computer*, vol. 41, no. 4, pp. 23–28, Apr. 2008.

[22] C. Borgelt, "An implementation of the FP-growth algorithm," in *Proc. ACM 1st Int. Workshop Open Source Data Mining, Frequent Pattern Mining Implement.*, 2005, pp. 1–5.

[23] J. Han *et al.*, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proc. 17th Int. Conf. Data Eng.*, 2001, pp. 215–224.

[24] G. M. Lee, H. Liu, Y. Yoon, and Y. Zhang, "Improving sketch reconstruction accuracy using linear least squares method," in *Proc. 5th ACM SIGCOMM Conf. Internet Meas.*, 2005, p. 24.

[25] M. Moh, S. Pininti, S. Doddapaneni, and T.-S. Moh, "Detecting Web attacks using multi-stage log analysis," in *Proc. IEEE 6th Int. Conf. Adv. Comput. (IACC)*, Feb. 2016, pp. 733–738.

[26] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proc. Conf. 23rd Eur. Symp. Artif. Neural Netw., Comput. Intell. Machine Learn. (ESANN)*, 2015, p. 89.

[27] Y. Yoon, H. Jung, and H. Lee, "Abnormal network flow detection based on application execution patterns from Web of things (WoT) platforms," *PLoS ONE*, vol. 13, no. 1, p. e0191083, 2018.

[28] G. Stoneburner, A. Y. Goguen, and A. Feringa, "Risk management guide for information technology systems," Comput. Secur. Res. Center, Tech. Rep. 800-30, 2002.

[29] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[30] S. Brin and L. Page, "Reprint of: The anatomy of a large-scale hypertextual Web search engine," *Comput. Netw.*, vol. 56, no. 18, pp. 3825–3833, 2012.

[31] N. Q. Zhu, *Data Visualization With D3.js Cookbook*. Birmingham, U.K.: Packt Publishing Ltd, 2013.

[32] X. Meng *et al.*, "MLlib: Machine learning in apache spark," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1235–1241, 2016.

[33] C. Gormley and Z. Tong, *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. Newton, MA, USA: O'Reilly Media, Inc., 2015.

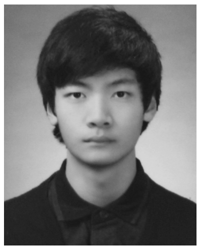
[34] J. Kreps *et al.*, "Kafka: A distributed messaging system for log processing," in *Proc. NetDB*, 2011, pp. 1–7.

[35] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. HotCloud*, 2010, p. 10.

[36] B. Hindman *et al.*, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proc. NSDI*, 2011, pp. 1–14.

[37] C. Kohlschütter, P.-A. Chirita, and W. Nejdl, "Efficient parallel computation of pagerank," in *Proc. Eur. Conf. Inf. Retr.*, 2006, pp. 241–252.

YOUNG YOON received the B.A. and M.S. degrees in computer sciences from the University of Texas at Austin in 2003 and 2006, respectively, and the Ph.D. degree in computer engineering from the University of Toronto in 2013. He has various technical and research experiences at Samsung Electronics, IBM T.J. Watson Research Center, Platform Computing, and Telus. He is currently an Assistant Professor in computer engineering with Hongik University, Seoul, South Korea. He is also the Director of Research at Netcore Tech Inc. One of the key projects he leads at Netcore Tech is designing a novel cyber security monitoring and control platform based on big data analytics and cloud for the Korean Ministry of Education.



YONGJUN CHOI received the bachelor's degree in computer engineering from Hongik University, South Korea, in 2016, where he is currently pursuing the master's degree in big data analytics and visualization. He is also a Graduate Research Assistant with the Application Platform Lab.