Interdependent Security Risk Analysis of Hosts and Flows

Mohsen Rezvani, *Student Member, IEEE*, Verica Sekulic, Aleksandar Ignjatovic,
Elisa [illegible], and Sanjay Jha, *Senior Member, IEEE*

***Abstract*—Detection of high [illegible] continues to be a significant problem i[illegible] high throughput networks. A compre[illegible] method should consider the risk propaga[illegible] flows. In this paper, this is achieved by [illegible] concepts. First, an interdependency relationship among the risk scores of a network flow and its source and destination hosts. On the one hand, the risk score of a host depends on risky flows initiated by or terminated at the host. On the other hand, the risk score of a flow depends on the risk scores of its source and destination hosts. Second, which we call flow provenance, represents risk propagation among network flows which considers the likelihood that a particular flow is caused by the other flows. Based on these two concepts, we develop an iterative algorithm for computing the risk score of hosts and network flows. We give a rigorous proof that our algorithm rapidly converges to unique risk estimates, and provide its extensive empirical evaluation using two real-world data sets. Our evaluation shows that our method is effective in detecting high risk hosts and flows and is sufficiently efficient to be deployed in the high throughput networks.**

***Index Terms*—Network risk assessment, flow provenance, risk propagation.**

I. Introduction

A SIGNIFICANT challenge for monitoring large enterprise networks is the difficulty of extracting risky network flows (the most likely malicious flows)[1] from a large number of flows. Identifying risky flows makes taking effective countermeasures feasible. For example, detected high risk network traffic can be forwarded to an Intrusion Detection System (IDS) for deep inspection in order to determine whether actual intrusions or other attacks are underway. The security risk score of a network flow can be evaluated based on both the risk score of the content of the flow and the risk of its related flows. Such recursive assessment complicates the detection of risky flows as it requires an accurate identification of the relationships among network flows.

Distributed attacks, such as distributed denial of service (DDoS) attacks and Botnet initiated attacks, are examples of attacks which generate malicious network flows that can be identified using the above recursive assessment. In such attacks, an attacker can exploit many methods and vulnerabilities across different systems in the network. A promising solution for risk assessment is one which takes into account inter-flow relationships. An illustrations of the inter-flow relationship is when an attacker creates a web session on a public web server hosted within a demilitarized zone (DMZ) by compromising the web server. Since the web related traffic to this server is permitted by the security policy, the attacker can initiate a new connection from the compromised server to another server in a protected network zone which allows the attacker to download rootkits or scan ports on the second server. This illustrates the fact that, in order to adequately evaluate the risk of the initial flow, we need to consider the whole interdependency risk relationship among recorded network flows.

This indicates that, to address the problem of flow risk assessment, we need a comprehensive solution which considers the whole interdependency risk relationship among network flows and the hosts initiating and targeted by the flows. The principle underlying such a solution is that the more risky flows a host initiates or is targeted by, the higher the risk score for the host. The risk score of a network flow partially depends on the risk scores of its source and destination hosts. Therefore, there is an interdependency relationship between network flows and hosts with respect to the assessment of their risk scores. The idea of employing link analysis techniques, such as PageRank and HITS [1], for detecting relevant flows has been recently proposed [2]. However, in such an approach the risk scores of hosts and flows are evaluated separately, without considering their interdependency. This type of risk assessment requires the evaluation of two separate dependency graphs for assessing the risk score of hosts and flows, which is very inefficient for high throughput networks.

In our recent work [3] we proposed an interdependency risk model for ranking the risk score of network flows as well as of the related hosts. In the proposed flow risk analysis, a network flow is likely to be risky if it is initiated or targeted by risky hosts. We consider a host to be risky if some of its related flows are risky. With such interdependency in mind, we develop an iterative algorithm for computing the risk score

Manuscript received February 22, 2015; revised May 13, 2015 and June 16, 2015; accepted June 24, 2015. Date of publication July 10, 2015; date of current version September 1, 2015. This work was supported in part by ICT Centre, CSIRO, Australia. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Jianying Zhou.

M. Rezvani, V. Sekulic, A. Ignjatovic, and S. Jha are with the School of Computer Science and Engineering, University of New South Wales and Purdue University, Sydney NSW 2052, Australia (e-mail: mrezvani@cse.unsw.edu.au; vericas@cse.unsw.edu.au; ignjat@cse.unsw.edu.au; sanjay@cse.unsw.edu.au).

E. Bertino is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA (e-mail: bertino@cs.purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2015.2455414

[1] In this paper, the term *risky flow* refers to a flow which is probably malicious and which is assigned a high risk score by a risk assessment method.

of hosts and flows. We consider two different aspects that may influence the risk score of a flow: the risk of flow attributes and the risk of the flow provenance. The risk of flow attributes is defined by an aggregation of the risk of the source and destination hosts of a flow and a predefined risk of the flow. We also define the risk of flow provenance which measures the amount of risk can be propagat[illegible]elated flows.

Extending our previous wo[illegible]ents a comprehensive risk assessmen[illegible] hosts with an extensive analytical a[illegible] With respect to our previous work[illegible] some significant new contributions.[illegible]nt the Flow Dependency Graph (FD[illegible]ighted flow causality relationship wh[illegible]lation of a weighted risk propagation model. Second, we provide a mathematically rigorous analysis of the behaviour of our iterative algorithm; in particular, its convergence and uniqueness of the solution are formally proved. Third, we study the time complexity and memory usage of our risk assessment algorithm, which are important efficiency factors for online monitoring algorithms. The experimental evaluation, conducted over two public datasets, confirms our analysis regarding the efficiency of the algorithm.

The reminder of the paper is organised as follows. The preliminary definitions are introduced in Section II. Section III presents the details of our risk computation model. Section IV presents the properties of our iterative algorithm. Experimental results are presented in Section V. We discuss potential evasion methods and their solutions in Section VI. Related work is discussed in Section VII and concluding remarks are made in Section VIII.

II. Basic Notions

In this section, we introduce several concepts used in our risk computation model.

A. Flow Causality

In a distributed system, an attack can be carried out by producing a sequence of network flows in which the first flow in the sequence has originated at the attacker host and the last flow has as its destination the target victim of the attack. We aim to model the flow causality between two flows through the likelihood that one of them has triggered the other one. We use three categories of flow information to formulate this likelihood: timing information, source and destination addresses, and a correlation among flows. The intuition behind the timing information is that a flow is more likely to be the cause of other flows if these flows started shortly after it [2]. Moreover, two flows can exhibit high correlation because a causal relationship exists between them [4]. Accordingly, we define the flow causality as follows:

Definition 1 (Flow Causality): The flow causality of flow f_y with respect to flow f_x, denoted as $\mathrm{fcs}(f_x, f_y)$, *is a measure of the likelihood that flow f_x triggered flow f_y, defined as:*

$$\mathrm{fcs}(f_x, f_y) = \alpha \times \mathrm{corr}(f_x, f_y) + (1-\alpha)e^{-\frac{|\mathrm{t}(f_x)-\mathrm{t}(f_y)|}{T}} \qquad (1)$$

where $\mathrm{t}(f_x)$ *denotes the start time of flow f_x and* $\mathrm{corr}(f_x, f_y)$ *is the correlation coefficient between features of f_x and f_y. T is the maximum distance between the starting times of two flows to be considered for causality relationship, and is called causality time interval. Moreover, α is a constant, $0 \le \alpha \le 1$ chosen to reflect the effect of the flow correlation in the flow causality, and is called causality factor.*

To take into account the time distance between two flows for computing the causality weight, we choose an exponential function which decreases sharply as time distance increases. While we could use other decay functions, our experiments indicate that using significantly slower decaying functions increases false positives. Using the exponential function with an adjustable parameter T appears to give enough flexibility for fine tuning the trade-off between sensitivity and false positives. Note that we limit our flow causality, as flow f_y is likely caused by flow f_x if during a particular causality time interval T, f_y starts after f_x and the source address of f_y is the same as the destination address of flow f_x. We have considered using other flow features to determine flow causality. However, unlike flow correlation, which benefits from multiple flow features, our experiments show that adding other features for flow causality, such as flow duration, does not improve the performance.

To compute the correlation between two flows, we first extract a feature vector from each flow, and then compute the absolute value of the Pearson correlation coefficient [5] between the feature vectors. Since this part is not the main focus of this paper, we include more details about the feature extraction and correlation computation in Appendix A.

B. Flow Dependency Graph

The idea behind the Flow Dependency Graph (FDG) is to model causality among flows in order to measure the propagation of risk across flows. This graph is based on the notion of graph proposed in [2].

Definition 2 (Flow Dependency Graph): An FDG is a weighted directed graph, generated by a sequence of flows $F = \{f_1, f_2, \ldots, f_n\}$ monitored during a particular window. The nodes are the flows in F, while the edges represent flow causality between corresponding flows, with the weight of each edge given by the weight function $\mathrm{fcs}(f_i, f_j)$ *from Eq.* (1).

Using the FDG allows our risk assessment approach to represent the causality between network flows. This causality may result from an attack in which the attacker creates connection chains. It also shows that the risk score of a flow not only depends on the risk of its features but is also influenced by the risk of other flows caused by such a flow.

Lemma 1: An FDG is a Directed Acyclic Graphs (DAG).[2]

A significant challenge in maintaining an FDG is the scalability, due to possibly very large numbers of flows in high throughput networks. Thus, in Section III we propose an efficient algorithm for risk computation based on the FDG which makes our methodology scalable.

[2] For all proofs in this paper, see Appendix B.

TABLE I
NOTATIONS USED IN THIS PAPER

F	set of all flows in the current window
H	set of all hosts in the current window
w	window length in number of flows
T	causality time inter[illegible]
$t(f)$	start time of a flow [illegible]
$\mathrm{fcs}(f_i, f_j)$	flow causality betw[illegible]
α	causality factor
$\mathrm{src}(f)$	source host of a flo[illegible]
$\mathrm{dst}(f)$	destination host of [illegible]
t_f	flow provenance o[illegible]
$\widehat{\mathrm{hr}}(h)$	intermediate risk s[illegible]
$\mathrm{hr}(h)$	final risk score of a host h for a window
$\mathrm{dr}(f)$	direct risk of a flow f
$\mathrm{pr}(f)$	risk score of provenance of a flow f
$\mathrm{npr}(f)$	normalised risk score of provenance of a flow f
$\widehat{\mathrm{fr}}(f)$	intermediate risk score of a flow f
$\mathrm{fr}(f)$	final risk score of a flow f for a window
$\mathrm{spr}(p)$	risk score of a risky path p
$N_p(f)$	number of risky paths in t_f
$N_t(f)$	number of flow causalities in risky paths of t_f

C. Flow Provenance

We first introduce the definition of risky path in an FDG, to be used to represent a potential attack scenario initiated by a flow.

Definition 3 (Risky Path): A risky path of a flow f is every path in the FDG starting at the node corresponding to f and ending at a leaf node in the graph.

In the risk evaluation algorithm, we will consider the risk score of risky paths for each flow. There may be more than one risky path for a flow in the FDG. Thus, the flows which participate in more risky paths will be assigned a higher risk score. We now introduce the notion of *flow provenance* to simplify the evaluation of the risk score for a flow.

Definition 4 (Flow Provenance): The flow provenance t_f of a flow f is a subgraph of the FDG with two properties: (1) t_f includes the node corresponding to f and all nodes which are included in the risky paths of f; (2) t_f contains all the edges of the FDG that connect two nodes in t_f.

Provenance in data trustworthiness models [6] represents the path taken in a network to transmit a data item from the source node to the destination node. By contrast, in our context, the notion of flow provenance for a flow f refers to the set of flows which have probably been originated by f. Therefore, the direction of flow provenance is somewhat reversed, in the sense that the risk of a flow is affected by the risk of further flows which are caused by it.

III. PROVENANCE-AWARE RISK COMPUTATION

In this section, we first describe the threat model and conceptual framework of our risk assessment. We then explain the details of its computation operations. A summary of notations is presented in TABLE I.

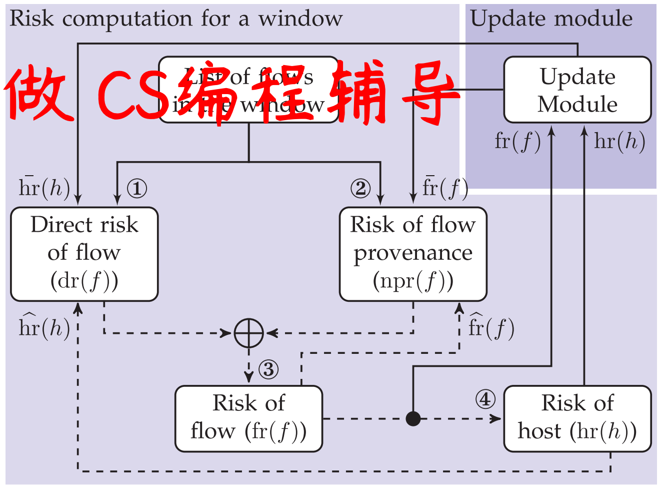

Risk computation for a window
Update module
List of flows in the window
Update Module
$\mathrm{fr}(f)$
$\mathrm{hr}(h)$
$\widehat{\mathrm{hr}}(h)$
①
②
$\widehat{\mathrm{fr}}(f)$
Direct risk of flow ($\mathrm{dr}(f)$)
Risk of flow provenance ($\mathrm{npr}(f)$)
$\widehat{\mathrm{hr}}(h)$
$\widehat{\mathrm{fr}}(f)$
③
④
Risk of flow ($\mathrm{fr}(f)$)
Risk of host ($\mathrm{hr}(h)$)

Fig. 1. Our iterative risk assessment framework.

A. Threat Model and Assumptions

We assume that an attacker aims to inject malicious flows into a target victim host for purposes such as: spreading malware, sending deceitful data or launching a DoS attack. The attacker uses connection chains among the hosts to hide his true origin. An example of steps in such a connection-chain attack is: (1) an attacker compromises a web server; (2) from the web server, he connects to a vulnerable internal host; and (3) an attack on the victim is then launched from the host over an internal connection. We assume that the attacker is able to: (a) utilize encryption and other methods to obfuscate the steps in the connection chain (for example using SSH protocol); and (b) to defeat deep packet inspection tools. We also assume that the adversary knowledge for the risk assessment methodology is limited. In Section VI we discuss mitigations techniques to be applied that allow us to remove this assumption.

B. Iterative Framework

In the proposed risk computation method, the risk scores are assigned to both hosts and flows, in an interdependent manner. Accordingly, the risk of a host is computed by aggregating the risk scores of the network flows which are either initiated by or targeted to the host. Furthermore, the risk score of a flow is measured by the risk scores of source host, destination host, provenance, and of a prior risk of the flow.

We assume that the monitored flows are an input stream for our system and they are handled in overlapping windows. Thus, we apply our risk assessment method on the current window. Moreover, the initial risk for hosts and flows are the risk scores obtained from the previous window.

Fig. 1 shows our iterative framework for computing the risk scores of hosts and flows. We first explain the overall architecture of our system, leaving the details to the subsequent sections. As shown in this figure, two main modules of our framework are the risk evaluation component for the current window and the update mechanism. Dashed lines are traversed in each iteration within a computation for each window; the solid lines are traversed between two consecutive windows.

The risk score of a flow is defined as an aggregate of its direct and provenance risk [illegible] F[illegible] ow of flows, we iteratively compute [illegible] e risk scores for the flows. In each [illegible] puted risk scores for each flow are [illegible] e new value of the risk score of the [illegible] equent iteration to update the risk sc[illegible] nd to compute the risk scores of th[illegible] of the host is then used in the next it[illegible] t risk scores.

Such iterative risk computations for the current window is repeated until the changes in the risk scores between two consecutive iterations become negligible. After completing the iterative risk computation, the risk scores of the hosts and the risk scores of the flows are transmitted to the update module. The update module then combines the current results with the results from the previous window to produce a weighted sum of such values. We explain the details of such computation process in the next sections.

C. Risk Score Computation for Network Flows

The risk score of each network flow is obtained by aggregating its direct and provenance risk score.

1) Direct Risk of Flow: The direct risk score of a flow is calculated from the risk scores of source host, destination host, and prior knowledge about its riskiness provided by the administrator. For the risk computation of source and destination hosts, we use the current risk score of the hosts, while for prior risk, we need to quantify prior knowledge about risky activities in the network. Having obtained all these three risk scores for a network flow f, we can define its direct risk score as a simple average as follows:

$$\mathrm{dr}(f) = \frac{\mathrm{hr}(\mathrm{src}(f)) + \mathrm{hr}(\mathrm{dst}(f)) + \mathrm{prior}(f)}{3} \tag{2}$$

where $\mathrm{src}(f)$, $\mathrm{dst}(f)$ and $\mathrm{prior}(f)$ denote the source, destination and prior risk score of flow f, respectively.

Note that using prior knowledge of riskiness of network activities is a common assumption in risk assessment systems [7]–[10]. The only constraint is that the prior risk scores must be normalised to the range of [0, 1]. This constraint helps to prove the convergence of our iterative algorithm. We use a simple quantification method to obtain the initial risk scores in our evaluation and also discuss how this assumption is dealt with in the literature (see Section V-A3).

2) Risk of Flow Provenance: As discussed, a flow provenance is a subgraph of the FDG, which includes a number of risky paths. The risk score of a risky path is obtained as a weighted sum of the risk scores of the flows within the path, with weight equal to the flow causality between the node and its parent in the risky path. The rationale behind such risk computation for a risky path is that the risk score indicates the likelihood of an attack proceeding via that risky path. Thus, the risk score of a risky path p, denoted as $\mathrm{spr}(p)$, is obtained as:

Algorithm 1 Recursive Computation of N_t and N_p

```
Input: $G$: an FDG, $v$: an boolean vector set by $false$
Output: vectors $N_t$ and $N_p$
procedure COMPUTENTANDNP(Graph $G$, Vertex $v$)
    if not visited[$v$] then
        $visited[v] \leftarrow true$
        Let $f_1, \ldots, f_k$ be $k$ child nodes of $v$
        if $v$ is a leaf node then
            $np[v] \leftarrow 1$ and $nt[v] \leftarrow 0$
        end if
        for all $i$ $(i = 1, \ldots, k)$ do
            if not visited[$f_i$] then
                COMPUTENTANDNP($G$, $f_i$)
            end if
            $np[v] \leftarrow np[v] + np[i]$
            $nt[v] \leftarrow nt[v] + np[i] + nt[i]$
        end for
    end if
    return $np$ and $nt$
end procedure
```

$$\mathrm{spr}(p) = \sum_{f \in p} \mathrm{fr}(f) \times \mathrm{fcs}(\mathrm{parent}(f, p), f). \tag{3}$$

Moreover, the risk score of flow provenance t_f is obtained as the sum of all risk scores of the risky paths within t_f, i.e.,

$$\mathrm{pr}(f) = \sum_{p \in t_f} \mathrm{spr}(p); \tag{4}$$

To ensure rapid convergence of our iterative risk computation algorithm, we normalise the risk score of flow provenances to a range of [0, 1]. To obtain such a normalisation, we first define the number of risky paths in provenance of flow f, denoted by $N_p(f)$ as:

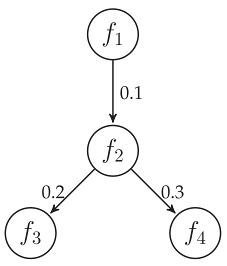
$$N_p(f) = \begin{cases} 1 & f \text{ is a leaf node,} \\ \sum\limits_{f \to f'} N_p(f') & \text{otherwise.} \end{cases} \tag{5}$$

Now, we compute recursively the number of flow causalities in all risky paths of the flow provenance t_f, denoted by $N_t(f)$, as:

$$N_t(f) = \begin{cases} 0 & f \text{ is a leaf node,} \\ \sum\limits_{f \to f'} (N_t(f') + N_p(f')) & \text{otherwise;} \end{cases} \tag{6}$$

where each edge is counted with a multiplicity equal to the number of risky paths containing that edge. The largest value of N_t in the graph is defined as $M = \max\{N_t(f') : f' \in F\}$. Given the above equations, the normalised sum of the right hand side of (4), i.e., $\sum\limits_{p \in t_f} \mathrm{spr}(p)/M$, can be written in a recursive manner as

$$\mathrm{npr}(f) = \begin{cases} 0 & f \text{ is a leaf node,} \\ \sum\limits_{f \to f'} \mathrm{npr}(f') + \dfrac{N_p(f')\mathrm{fr}(f')\mathrm{fcs}(f, f')}{M} & \\ & \text{otherwise.} \end{cases} \tag{7}$$



$\mathrm{fr}(f_2) = 0.4, \mathrm{fr}(f_3) = 0.5, \mathrm{fr}(f_4) = 0.6.$
$N_p(f_1) = N_p(f_2) = 2, N_p(f_3) = N_p(f_4) = 1.$
$N_t(f_1) = 4, N_t(f_2) = 2, N_t(f_3) = N_t(f_4) = 0.$
$M = 4.$
Now, using Eq. (7), we have:
$\mathrm{npr}(f_3) = \mathrm{npr}(f_4) = 0$
$\mathrm{npr}(f_2) = \frac{0.5\times0.2+0.6\times0.3}{4} = 0.07$
$\mathrm{npr}(f_1) = 0.07 + \frac{2\times0.4\times0.1}{4} = \mathbf{0.09}$
The graph contains two risky paths:
$p_1 = f$ … f_4
Now …):
$\mathrm{spr}(p_1$
$\mathrm{spr}(p_2$
$\mathrm{pr}(f_1)$

Fig. 2. Example of risk computation f… on the edges of the FDG are the flow causali… the right side shows the risk computation for p… using both (7) and (4). As it is shown, these … lues for the risk scores, while using Eq. (7) w… core of provenance for all flows in the FDG.

Algorithm 2 Recursive Algorithm for Computing Risk of Flow Provenance

```
Input: $G$: an FDG, $v$: a vertex in the FDG
Output: npr: Risk of provenance for all flows in the FDG
procedure ProvenanceRisk(Graph $G$, Vertex $v$)
    if not visited[$v$] then
        $visited[v] \leftarrow true$
        Let $f_1, \ldots, f_k$ be $k$ child nodes of $v$
        if $v$ is a leaf node then
            $npr[v] \leftarrow 0$
        end if
        for all $i$ $(i = 1, \ldots, k)$ do
            if not visited[$f_i$] then
                ProvenanceRisk($G$, $f_i$)
            end if
            $npr[v] \leftarrow npr[v] + npr[i] + (np[i] \times \mathrm{fr}(f_i) \times \mathrm{fcs}(f_v, f_i))/M$
        end for
    end if
    return $npr$
end procedure
```

Since $N_t(f)$ and $N_p(f)$ depend only on the structure of the graph, for each window we compute $N_t(f)$ and $N_p(f)$ once, before starting the iterative risk computation procedure. We use a Depth First Search (DFS) algorithm to compute these values, as shown in Algorithm 1.

Eq. (7) shows that the risk score of the flow provenance of a network flow depends on the risk scores of its children in FDG. Thus, during the recursive procedure for computing the risk scores, we use, in each round of iteration, a DFS algorithm to compute the normalised risk scores of provenances for all the flows in the graph, as shown in Algorithm 2. An example of the risk computation for a simple FDG is shown in Fig. 2.

3) Flow Risk Aggregation: As described, the risk score of a flow is computed by aggregating its direct and provenance risk scores (see ③ in Fig. 1) by a weighted sum:

$$\widehat{\mathrm{fr}}(f) = c_f \mathrm{dr}(f) + (1 - c_f)\mathrm{npr}(f) \tag{8}$$

where c_f is a constant, $0 \leq c_f \leq 1$. Thus, an administrator can select a larger value for c_f when confident which activities are risky. On the other hand, a smaller value for this constant is appropriate when a higher weight is to be given to flow provenances.

D. Risk Score for Hosts

The risk score of a host is computed from its engagement in risky incoming and outgoing flows (see ④ in Fig. 1). Incoming and outgoing risky flows have different effects which must be considered in the computation process. For example, network flows that are initiated by a web server inside a DMZ have more influence on the risk score of the server than incoming flows to the web server. In contrast, incoming flows to a internal desktop host have more effect on the risk score of the host than its outgoing flows. Therefore, we allow the network administrator to manipulate this impact factor based on the network topology. We propose the following equation:

$$\widehat{\mathrm{hr}}(h) = \frac{c_{in} \sum_{f \in F_{I,h}} \mathrm{fr}(f)}{|F_{I,h}|} + \frac{(1 - c_{in}) \sum_{f \in F_{O,h}} \mathrm{fr}(f)}{|F_{O,h}|} \tag{9}$$

where $F_{I,h}$ and $F_{O,h}$ are the set of incoming and outgoing flows of host h, respectively (in the current window), and $|F|$ is the cardinality of set F.

In Eq. (9), c_{in} is a constant in the range $0 \leq c_{in} \leq 1$ chosen to reflect the impact of incoming flows in the computation of the risk score. For example, if c_{in} has a large value, especially if $c_{in} > 0.5$, we consider the incoming flows to be more risky than the outgoing flows for the host risk computation. In this case a higher proportion of the risk score of a flow is propagated to its destination host. Such larger values of c_{in} can be used, for example, for detecting victim hosts of an attack. On the other hand, if c_{in} has a smaller value, especially if $c_{in} < 0.5$, a high proportion of the risk score of a flow is propagated to its source host. Thus, our risk computation method will assign a high value of risk to the attackers hosts. In summary, if c_{in} is larger, the targeted hosts will be assigned higher values of risk; in contrast, for smaller c_{in}, originators will be assigned higher values of risk.

E. Iterative Algorithm

As explained, an iterative algorithm is employed to compute the risk scores for flows and hosts within each window. Algorithm 3 shows the iterative process; a host and a flow risk vectors (respectively **hr** and **fr**) are inputs from the previous window. The algorithm has as input a set F of monitored flows for the current window. As we will show, our algorithm converges to a unique solution; however, passing the risk values to the next window greatly reduces the number of iterations till convergence.

F. Update Process

In the update process and before starting the risk computation for the next window w_{i+1}, we first obtain initial risk scores $\bar{\mathrm{hr}}_{i+1}(h)$ for each host h and $\bar{\mathrm{fr}}_{i+1}(f)$ for each flow f, to be used as initial values for the computation

Algorithm 3 Iterative Risk Computation in Each Window

```
Input: hr, fr: host and flow risks from the previous window,
    F: list of flows in the current window
Output: hr, fr: updated risk values
procedure RiskComputation(hr, fr, F)
    Let H the set of hosts within F
    Create FDG G from F
    ComputeNtAndNp(G
    M ← max{N_t(f) : f ∈
    repeat
        Compute dr(f) for al
        Compute npr(f) for a                                  2
        Compute \widehat{fr}(f) for all
        Compute \widehat{hr}(h) for al
        fr ← \widehat{fr}
        hr ← \widehat{hr}
    until the change of risk scores is negligible
    return hr and fr
end procedure
```

of window w_{i+1}. These risk scores are provided by the update process, as a weighted sum of the corresponding values $\text{hr}_{w_i}(h)$, from the previous window w_i, and $\bar{\text{hr}}_i(h)$ from the previous values obtained by the update module for each host h. The initial risk scores $\bar{\text{fr}}_{i+1}(f)$ of each flow f, from the corresponding values $\text{fr}_{w_i}(f)$ and $\bar{\text{fr}}_i(f)$, are obtained in the same manner:

$$\bar{\text{hr}}_{i+1}(h) = c_{hu}\text{hr}_{w_i}(h) + (1 - c_{hu})\bar{\text{hr}}_i(h) \tag{10}$$

$$\bar{\text{fr}}_{i+1}(f) = c_{fu}\text{fr}_{w_i}(f) + (1 - c_{fu})\bar{\text{fr}}_i(f) \tag{11}$$

where c_{hu} and c_{fu} are constants, with $0 \leq c_{hu}, c_{fu} \leq 1$, which determine the relative importance of values from the current window versus previous update values.

IV. Properties of the Algorithm

In this section, we highlight relevant properties of the iterative algorithm including convergence, uniqueness, and memory and time complexities.

A. Risk Discrepancy Bounds

Lemma 2: The risk score of provenance, flows and hosts is always in the range [0, 1].

Corollary 1: The maximum difference between the risk scores of either a flow or a host between any two iterations is 1.

Proof: Follows immediately from the fact that all risk scores are in the range of [0, 1]. □

Lemma 3: The difference of the risk scores of any flow f and any host h obtained at two consecutive iterations t and $t+1$ is bounded by an exponential function of t:

$$\left|\text{fr}^{(t+1)}(f) - \text{fr}^{(t)}(f)\right| \leq \left(1 - \frac{1}{3}c_f\right)^t \tag{12}$$

$$\left|\text{hr}^{(t+1)}(h) - \text{hr}^{(t)}(h)\right| \leq \left(1 - \frac{1}{3}c_f\right)^t \tag{13}$$

B. Proof of Convergence

Using the above risk discrepancy bound, it can be proved that the risk scores for both flows and hosts converge. Moreover, it can also be shown that our algorithm converges rapidly.

Theorem 1: For every flow $f \in F$, sequence $\{\text{fr}^{(t)}(f)\}$, $t \in \mathbb{N}$, and for every host $h \in H$, sequence $\{\text{hr}^{(t)}(h)\}$, $t \in \mathbb{N}$, converge.

Lemma 4: For every value $\varepsilon > 0$ of the threshold, Algorithm 3 converges after $N = \delta + \log_{(1-\frac{1}{3}c_f)} \varepsilon$ many steps.

C. Proof of Uniqueness

In this section, we prove that algorithm 3 provides unique solutions for risk scores of both hosts and flows.

Theorem 2: For any given assignment of the prior risk scores for every flow $f \in F$, sequence $\{\text{fr}^{(t)}(f)\}$, $t \in \mathbb{N}$, and for every host $h \in H$, sequence $\{\text{hr}^{(t)}(h)\}$, $t \in \mathbb{N}$, converge to unique values, independent on the initial values of $\text{fr}^{(0)}(f)$ and $\text{hr}^{(0)}(h)$.

D. Memory Usage and Complexity Analysis

The memory required by the algorithm mainly consists of the memory needed for storing of all risk scores for hosts and flows in the window as well as the temporary values used for computing the risk of flow provenances, which is in $O(w)$, where w is the window length. Also, the space complexity of Algorithm 2 is in $O(w)$ as it is a DFS-like algorithm. Moreover, the space complexity for maintaining a FDG is in $O(|V| + |E|)$ using an adjacency matrix where $|V|$ and $|E|$ are the number of nodes and edges in the graph, respectively. Since the maximum number of edges in a DAG is $\frac{|V|\times(|V|-1)}{2}$, the space complexity of Algorithm 3 in the worst-case is in $O(w^2)$.

The time complexity of the risk computation for all flows, hosts, and the aggregation of the direct risk of flow and flow provenance in a single iteration is in $O(w)$. The time complexity of Algorithm 2 for computing the risk of flow provenance is in $O(|V| + |E|)$ as it is a DFS-like algorithm. Thus, each iteration of our algorithm in worst case requires $O(w^2)$ steps, and thus for k iterations, the total running time for the iterative algorithm is in $O(k \times w^2)$. The fact that the total number of iterations is logarithmic in the value of the threshold guarantees the efficiency of our method. Moreover, since the FDG is sparse, both space and time complexities of our approach can be reduced in a linear to the number of flows.

V. Experimental Evaluation

A. Experimental Environment

We evaluate our method using using two public datasets including two attacks. To evaluate the effectiveness, we perform our risk computation on both these datasets and show that our method assigns high risk scores to the victims and attackers which are involved in the attacks. To evaluate the efficiency, we measure the elapsed time for our approach and two other recent models based on PageRank and HITS [2]. All the experiments were conducted on an iMac PC with

TABLE II
EXPERIMENTAL PARAMETERS

<table>
<tr><td colspan="3">Causality time interval (seconds)</td><td>$T = 600$</td></tr>
<tr><td colspan="3"># of flows within a window</td><td>$w = 2,000$</td></tr>
<tr><td colspan="3">Sliding percentage of the window</td><td>35%</td></tr>
<tr><td colspan="3">The maximum input cache size (# of flows)</td><td>10,000</td></tr>
<tr><td colspan="3">Accuracy threshold in the iter[illegible]</td><td>[illegible]1</td></tr>
<tr><td colspan="3">Constants c_f, c_{hu} ar[illegible]</td><td>[illegible]</td></tr>
<tr><td>Constant c_{in}</td><td>0.75</td><td>[illegible]</td><td>[illegible]</td></tr>
</table>

2.00GHz Intel Core 2 Duo p[illegible] running Ubuntu 12.04 LTS. The [illegible]ten in Java with JDK 1.7. TABLE I[illegible]mental parameters.

1) Honeynet Dataset: To evaluate the effectiveness of our model, we apply it to the public traces captured by the Honeynet project, Scan 18 [11]. The attack within this dataset consists of scanning, compromising, downloading and installing a Rootkit, and sending spam emails. In this attack, the attacker compromises a local honeypot machine with IP address 172.16.1.108 using at least two different stepping stones to scan and attack the network. Moreover, the attacker exploits a number of different IP addresses including 211.185.125.124, 211.180.229.190, 193.231.236.41, 216.136.129.14, and 209.61.188.33.

2) MIT Lincoln Dataset: The MIT Lincoln dataset includes a DDoS attack and was originally created as part of a DARPA project for evaluating IDSs [12]. The data file consists of 103,006 flows and 34,511 hosts. The publicly available labeled list of flows shows that there are 33,816 malicious flows in this dataset. In this dataset, the attacker installs hacking tools on three machines inside the network with IP addresses 172.16.115.20, 172.16.112.50, and 172.16.112.10. After compromising these three victims, the attacker launches a DDoS attack to the victim machine 131.84.1.31 by flooding packets.

3) Prior Risk Assignment: According to the computation of direct risk given by Eq. (2), we allow the possibility that the prior risk scores of flows are supplied by the administrator, based on his prior knowledge of the network activities. In Section VI, we discuss some different methods proposed in the literature for quantifying prior knowledge.

In this paper, we use a simple method to quantify the prior risk, based on a publicly available list of malicious ports and the history of detected risky hosts obtained by our algorithm in previous windows. The reasons for choosing this method are: (1) it has been shown that the PageRank and HITS algorithms provide effective results using this method [2] and it therefore helps us compare the effectiveness of our algorithm with these algorithms; (2) we have no prior knowledge of risky activities in the network where the datasets has been collected.

We assign a higher risk score to a flow when either its destination port is in the risky ports listed by the Emsisoft Portlist [13] or its source or destination address is among the high risky hosts obtained in previous window. The Emsisoft Portlist [13] lists the TCP/UDP ports that are more frequently exploited by malwares, as well as the malwares using these ports. Thus, if the destination port of a flow is in this port list, we assign a higher risk score for the prior risk of that flow. We keep track of the high risky hosts obtained by our approach over each window, which will be used as a part of our prior risk assignment.

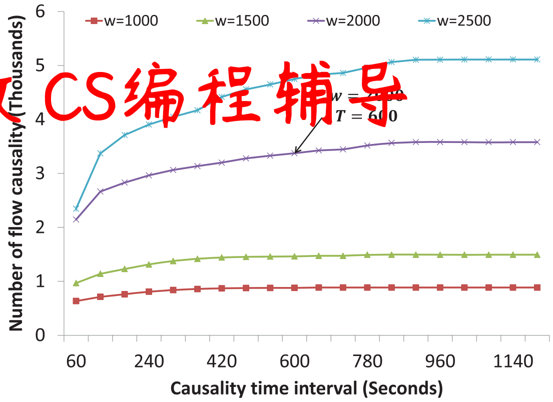


Fig. 3. Average number of flow causalities in windows.

B. Sensitivity Analysis

Beyond investigating the effectiveness and efficiency of our risk assessment approach, we also measure the sensitivity of the results with respect to the risk computation parameters: the length of the sliding window and the causality time interval (Section II-A). Both these parameters affect the efficiency of our approach because the number of flow causalities depends on them. The number of causalities identifies the number of edges in the FDG which is a significant parameter for the computational complexity of our risk computation algorithm, as shown in Section IV-D. Therefore, in this section we investigate the number of flow causalities according to different values for these two parameters by running experiments over a subset of the MIT Lincoln dataset including its first 10,000 network flows.

Fig. 3 compares the average number of flow causalities on the MIT Lincoln dataset with respect to both the sliding window length and causality time interval. We can see that by increasing the causality interval time, we achieve a stable number of flow causalities for each value of window length. The reason is that a very large value of the causality interval time exceeds the maximum time interval between two network flows within the window. Since the window length is based on the number of flows, we can estimate a maximum threshold for the causality time interval according to the window length and the throughput of the network.

Accordingly, an administrator must consider two points for choosing the value of the window length and causality time interval: (1) the window length can be selected based on the available memory and processing power. This is important because increasing the window length can improve the effectiveness of the risk assessment algorithm (as shown in Section V-E), (2) after selecting a large-enough window length, the administrator can select the causality time interval based on the throughput of the network. Choosing a large value for the causality time interval in a high-throughput network

TABLE III

RISK EVALUATION RESULTS FOR HONEYNET SCAN 18 DATASET TO ASSIGN HIGHER RANK TO THE ATTACKER HOSTS

	$c_{in} = 0.75$		$c_{in} = 0.90$	
Rank	Risk	Host	Risk	Host
1	0.30870509	**172.16.1.108**	0.30868912	**172.16.1.108**
2	0.2235409	*211.185.125.1*	[illegible]	**.103**
3	0.17689227	**172.16.1.10**	[illegible]	*5.124*
4	0.10196934	*193.231.236.*	[illegible]	*36.41*

makes the time interval ineffe[illegible] steady states for these large values i[illegible] small value of time interval opens t[illegible]anced attack goes undetected. We discuss possible evasion techniques in Section VI. Accordingly, we select values of $w = 2000$ and $T = 600$ for our evaluations.

C. Effectiveness for Honeynet Dataset

In this section we set c_{in} to 0.75 (0.25) to evaluate the effectiveness of our approach for detecting victim (attacker) hosts. Once our approach assigns a risk score to each host, we need a systematic mechanism to determine a cut-off (threshold) value which helps us to report a host with a risk score higher than the threshold, above which the host is labelled potentially malicious. The traditional method is to determine the cut-off value by using an expression of the form $\mu - \beta \times \sigma$, where μ and σ are the sample mean and sample variance of the risk scores, respectively. β is a threshold constant which can be adjusted by the administrator, based on the expected number of flows and the available computational resources. In our experiments we choose $\beta = 0.1$ to determine what hosts are deemed as *high risk hosts*.

High risk hosts in the Honeynet dataset as ranked by our tool are reported in TABLE III. The obtained cut-off value for these results is 0.054. In this table, the hosts indicated in bold (italic) are the actual victims (attacker) according to the ground truth for the dataset. It shows that our system assigns high risk scores to the main victim (172.16.1.108). Moreover, the third high risk host, 172.16.1.103, is the next victim which responded to the SYN scan of the attacker in the dataset [11]. The reason of the high risk score for the first host is that, in the attack, host 211.185.125.124 launched a sequence of a "RPC GETPORT Call" followed by the actual buffer-overflow attack, first to 172.16.1.103 and then to 172.16.1.108. Moreover, host 172.16.1.103 was immune from this attack while host 172.16.1.108 was affected by it [11]. Our approach is more effective compared to PageRank and HITS as neither the algorithms managed to rank the second victim host (172.16.1.103) as a risky host [2].

Moreover, we expect to see lower risk scores for attacker 211.185.125.124 compared to victim 172.16.1.108 as we set $c_{in} = 0.75$. This is because of the high proportion of malicious flows initiated by the attacker. To address this problem, we set $c_{in} = 0.90$ and the right part of TABLE III shows that our approach assigns the highest risk scores to both victims. It also show that our approach is effective for detecting attackers.

TABLE IV

RISK EVALUATION RESULTS FOR HONEYNET SCAN 18 DATASET TO ASSIGN HIGHER RANK TO THE VICTIM HOSTS

	$c_{in} = 0.25$		$c_{in} = 0.10$	
Rank	Risk	Host	Risk	Host
1	0.32280532	**211.185.125.124**	0.3613098	**211.185.125.124**
2	0.2948738	172.16.1.108	0.28765184	172.16.1.108
3	0.1184613	**193.231.236.41**	0.12197655	**193.231.236.41**
4	0.08737538	172.16.1.103	0.0669748	172.16.1.103
5	0.05484723	**211.180.229.190**	0.06597695	**211.180.229.190**

TABLE IV reports the high risky hosts ranked by our approach with two values for c_{in}: 0.25 and 0.10. The obtained cut-off value for these results is 0.055. It shows that the highest risk host detected by the new settings is the attacker which initiated most attack traffic. It is because that our tool propagates a higher proportion of risk of flows towards the hosts which initiated the flows. Accordingly, it can be seen that by decreasing the value of cin from 0.25 to 0.10, the risk scores of all three attacker hosts increase and also the risk scores of both victim hosts have decrease.

The results also show that two victims are still in the high risky hosts. This can be explained by the fact that most of the malicious flows in the dataset was initiated by host 211.185.125.124 and targeted both victims; this causes a high amount of risk to be propagated to the victim hosts in our approach. A possible solution for excluding such propagation may be inverting the link directions in the FDG. This can help us force the risk propagation from the risky flows toward the victims.

D. Effectiveness for Lincoln Dataset

In this experiment, we apply our tool to the MIT Lincoln dataset which contains 103,006 flows and using the parameters presented in TABLE II, the tool uses 79 windows. At the end of each window w_i, $1 \leq i \leq 79$, we have partial risk scores $\text{hr}_{w_i}(h)$ and $\text{fr}_{w_i}(f)$ for hosts and flows, respectively. In practice, such results, obtained in real time, would be used for monitoring risky activities.

Fig. 4 shows for each host in how many windows (out of 79) the host was ranked among the high risky hosts. In this experiment the obtained cut-off value is 0.011 and consequently 21 high risky hosts have been selected. We report the first 10 risky hosts in Fig. 4. The figure shows that our approach detected the main victim 131.84.1.31 and two compromised machines 172.16.115.20 and 172.16.112.50 among the risky hosts. The highest values of risk for address 131.84.1.31 in both experiments can be the result of many malicious flows targeted to the host during the DDoS attack [12].

An examination of the effectiveness of our approach and the effectiveness of the PageRank and HITS algorithms shows that none of these algorithms could detect the third victim 172.16.112.10 [2]. This is due that only 15 malicious flows were targeting this host and it would be very difficult to rank

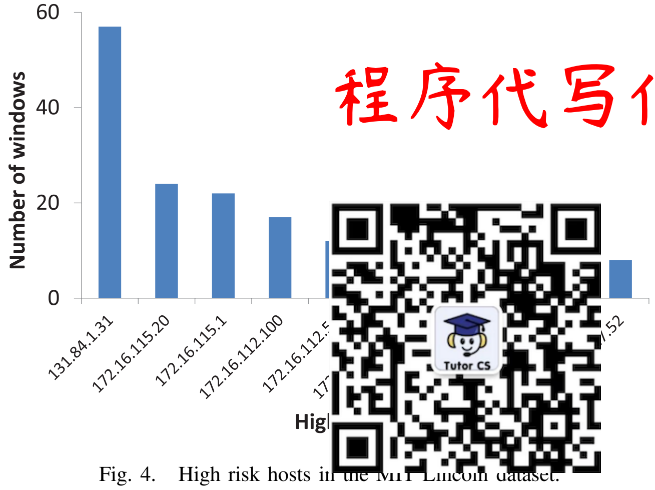


Fig. 4. High risk hosts in the MIT Lincoln dataset.

TABLE V

Confusion Matrix for Detecting Risky Flows

Actual Class	Predicted Class	
	Attack	Benign
Attack	True Positive (TP)	False Negative (FN)
Benign	False Positive (FP)	True Negative (TN)

the host as a high risky host using such a small number of malicious flows. Note that the attacker used many IP addresses to initiate the DDoS.

E. Malicious Flows Detection Performance

As discussed, the proposed risk assessment approach assigns risk scores to both hosts and flows. In this section, we evaluate the performance of the approach for ranking malicious flows using a publicly available labelled list of flows for the MIT Lincoln dataset [12]. The list contains the 5-tuple of all flows within the MIT Lincoln dataset along with a label which is either malicious or benign. Thus, we use this list as the ground truth for evaluating the performance of our method.

Since the main objective of our approach is to compute the risk of network flows, we need a method to evaluate the results of the risk assessment as a binary classification technique. To this end, we extract the k-top risky flows ranked by our approach in each window, where k is the number of malicious flows in that window according to the ground truth. Next, we compute the confusion matrix (as shown in TABLE V) for each window by comparing these two flow lists which helps us measure the true positive rate (TPR) and false positive rate (FPR) of the risk assessment methods as follows:

$$TPR = \frac{TP}{TP + FN} \times 100 \tag{14}$$

$$FPR = \frac{FP}{FP + TN} \times 100 \tag{15}$$

Note that we choose TPR and FPR because in our evaluation FP and FN are identical. This is because that the number of flows in the predicted classes are same as actual ones. Thus, TPR and Precision will be identical.

TABLE VI

True Positive Rate of the Algorithms

	Provenance	PageRank	HITS-Authority
w=2000	78.98	62.77	79.37
w=2100	81.63	56.15	100.0
w=2200	88.10	52.06	74.27
w=2300	97.47	56.15	62.99
w=2400	95.95	53.92	61.30
w=2500	97.81	45.23	63.56

TABLE VII

False Positive Rate of the Algorithms

	Provenance	PageRank	HITS-Authority
w=2000	0.60	1.16	0.76
w=2100	0.05	0.05	0.00
w=2200	2.00	2.80	1.68
w=2300	0.09	0.04	0.04
w=2400	0.17	0.21	0.21
w=2500	0.20	0.24	0.24

TABLE VI shows the maximum TPRs of the three risk assessment algorithms with respect to different window lengths. One can see that there is an increasing trend in our approach (referred to as *Provenance* in the table) as window length increases. The results show that the performance of our approach is superior to the performance of the other algorithms when increasing window length. The reason is that our algorithm has more chance to detect possible causality among the flows in the window. However, the performance of both *PageRank* and *HITS-Authority* often decreases as window length increases. This is due that these algorithms assign a very high risk to only a small cluster of risky flows. Therefore, while the number of malicious flows increases in the window, the TPR of the algorithm decreases. A similar observation is reported in [2] where the proportion of zero-ranked flows, which were assigned a risk score of zero, significantly increases as the number of flows increases. However, in our approach the risk score of a flow is computed based on both the risk of its provenance and its direct risk. Therefore, our risk propagation can prevent such a sharp increase leading to assign very high risk scores to a small cluster of risky flows and zero scores to most remaining flows.

The results also show that *HITS-Authority* has a high accuracy for the window length of 2100. However, it fluctuates as the window length increases. Note that the HITS algorithm produces two rank values: *authority* and *hub*. The hub values obtained in this experiment are identical to the values obtained by PageRank. Therefore, the high accuracy of *HITS-Authority* for point $w = 2100$ might be due to the fact that the risky flows had few outgoing edges which prevents them from propagating the obtained authority rank. This situation is due to a very prominent weakness of the HITS algorithm, called *link spamming*, which can lead to high authority and low hub values for risky flows [1].

TABLE VII shows the minimum FPR of the three risk assessment algorithms with respect to different

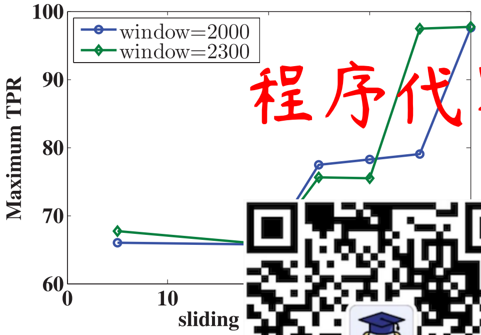


Fig. 5. True positive rate of our [illegible] sliding percentage of the flow window.

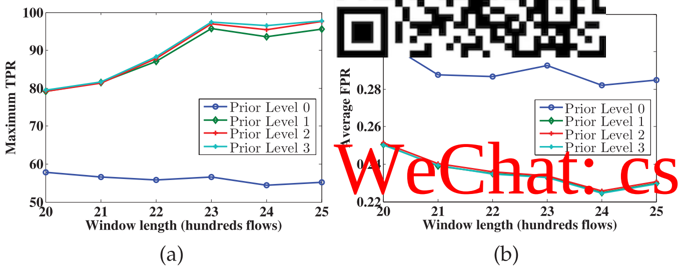


Fig. 6. TPR and FPR of our approach with respect to different prior knowledge levels. (a) Maximum TPR. (b) Average FPR.

window lengths. One can see that all three algorithms provide promisingly low FPR which validates their effectiveness for risk assessment. Similarly, our approach has lower FPR as the window length increases. The reason is that there is a direct relationship between TPR and FPR in our confusion matrix. The results also validate our claim that the iterative algorithms can effectively filter out the intrinsic false positives in our flow causality definition.

As described, our approach uses sliding windows to relate the possible causalities among the flows within two consecutive windows. In this experiment, we evaluate the impact of the sliding window percentage on the effectiveness of our approach. Fig. 5 shows the maximum TPRs of our risk assessment algorithm with respect to different values for the sliding percentage of the flow window. Notice the sharp increase of TPR at sliding percentage of 40% for both window lengths.

To investigate how prior knowledge can influence the effectiveness of our approach, we evaluate the system with four different levels of prior knowledge: (0) there is no prior knowledge and thus we assign identical prior risk to all flows, (1) the prior knowledge is only the public risky port list (see Section V-A.3), (2) we assign a higher prior risk score to flows either initiated by or targeting one of the top-10 risky hosts detected in the previous window, and (3) we assign a low risk score to destination ports 80 and 25, and a high risk score to destination ports higher than 1024. Note that each knowledge level includes the knowledge of the previous levels. Fig. 6 shows how the performance of our approach improves when better prior knowledge of risky activities is provided.

F. Efficiency

We quantify the efficiency of our tool by analysing its memory usage and processing time. The memory usage of the system is only dependent on the window length (see Section IV-D). Thus, we compare the processing time (the elapsed CPU time) of our provenance-aware algorithm with the PageRank and HITS based approaches [2] with respect to different window lengths. Here, we apply PageRank and HITS only to the FDG and then compute the elapsed times as the efficiency of the approaches.

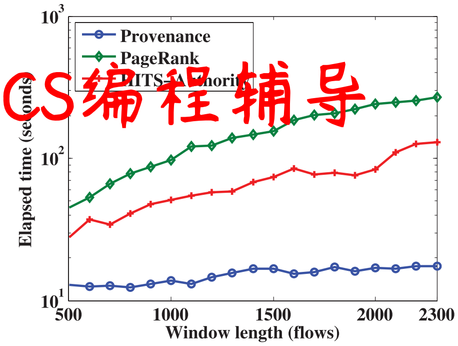


Fig. 7. Elapsed time of the algorithms.

Fig. 7 shows the elapsed time of our approach along with PageRank and HITS. One can see that the elapsed time of both the PageRank and HITS algorithms sharply increases as the window length increases (the green and red lines, respectively). This is because the size of the FDG directly depends on the size of the window and the high processing time is due to the application of the PageRank to a very large graph. Moreover, the PageRank algorithm runs on two graphs for computing the risk scores of hosts and flows. However, our approach computes both the scores of hosts and flows for such window in a reasonable processing time (the blue line).

We also investigate the average number of iterations needed for convergence for all three algorithms over the MIT Lincoln dataset. When limiting the algorithms to the same convergence threshold and for $w = 2000$, the average number of iterations for *Provenance* was 6, for *PageRank* was 19, and for *HITS-Authority* was 12. The very fast convergence in our algorithm is due that its number of iterations is logarithmic in the value of the convergence threshold (see Lemma 4 and Section V-G). This observation, along with the fact that saving just a handful of iterations in the link analysis algorithms is praiseworthy [1] given the significant gap between the elapsed time of our method and other two algorithms, explains why our method is definitely more efficient.

G. Analysis of Discrepancy and Convergence

In this section, we perform a set of experiments to analyze the properties of our iterative algorithm in terms of discrepancy and convergence. We investigate two types of discrepancies for the risk scores of both flows and hosts computed in each iteration of Algorithm 3 over the Honeynet Scan 18 dataset. For each of flow and host risk scores, we define the maximum discrepancy by choosing the worst-case discrepancy for all flows and hosts, respectively. Therefore, the maximum discrepancy at iteration l is computed as follows:

$$discrepancy^{(l)}_{\text{fr}} = \max\left\{\left|\widehat{\text{fr}}^{(\infty)}(f) - \widehat{\text{fr}}^{(l)}(f)\right| : f \in F\right\}$$

$$discrepancy^{(l)}_{\text{hr}} = \max\left\{\left|\widehat{\text{hr}}^{(\infty)}(h) - \widehat{\text{hr}}^{(l)}(h)\right| : h \in H\right\}$$

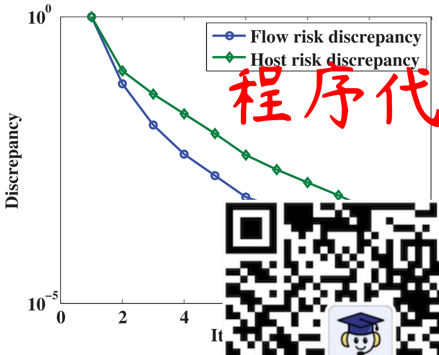


Fig. 8. Exponentially [illegible]

Fig. 8 shows that the afore[illegible] in our approach decline exponentially [illegible] ations increases. The results validate our analytical analysis for the convergence of our iterative algorithm presented in Lemma 4.

VI. Discussion

Like any network security mechanism, our risk assessment can be attacked. More specifically, as flow causality is based on both flow timing information and flow correlations, one possible attack is against the timing flow causality. In such an attack the attacker uses a large number of bots to flood a server by sending many requests at the same time. In this scenario, the attacker can evade the timing flow causality. However, in our approach, the dependency among the malicious flows in this attack can be detected using the flow correlation as the flows are highly correlated.

The attacker can also try to inject delays to compromise the timing flow causality. A possible solution to detect the causalities in such scenario can be increasing T. We believe that in such circumstances our system can rely on flow correlation as using slower decaying functions or increasing T excessively increases FPR.

Another possible attack is one by which an attacker bypasses the well-known malicious ports by using random port numbers or legacy ports associated with well-known and benign applications. Although using the simple method of quantifying the prior risk assignment shows promising accuracy in our experiments, we believe that a network administrator can provide more effective prior risk scores for the network activities using his knowledge about the network. For example, the network administrator can determine the prior risk based on either the previous detection mechanisms (as we used in our experiments) or external blacklists [14].

In addition to attacks, our approach can be prone to false positives in risk assessment, even when considering both timing information and flow correlation. The reason is the intrinsic correlation that exists among flows, even when they are not related [15]. While this is indeed the case, our iterative procedure is likely to filter out such false positives (as shown in Section V-E), unless false positives happen on the entire risky path, which is unlikely. Moreover, we should remember that such unlikely events are acceptable, because we aim at identifying flows which are *likely* to be risky. Having isolated a relatively small number of high risk flows, one can supplement our method with other means of flow dependency analysis, such as active traffic analysis methods, recently suggested to address the false positives by injecting invisible tags into the traffic [15].

A critical element in the accuracy of our method is represented by the FDG as risk scores are propagated through the FDG. Hence, the definition of the graph is of critical importance for method to be successful, as different graphs can yield different results. We have leveraged the flow causality to create the FDG's edges and have shown the effectiveness of our approach for assigning high risk scores to victims and attackers. One may define a different dependency graph or edge weighting strategy, as they are application-specific. Given such a graph and some domain knowledge about the prior risk (i.e. the blacklist), our method can be directly used.

VII. Related Work

The work most closely related to our research is by Wang *et al.* [2], which aims at risk assessment for hosts and flows by separately employing the PageRank and HITS algorithms. Risk assessment methodologies have been recently investigated by applying the idea of belief propagation on graphs representing networks. Feng and Li [10] propose an information risk assessment method which leverages a fuzzy belief assignment based on the Dempster-Shafers theory. Coskun *et al.* [16] propose an iterative belief propagation method on a mutual contacts graph in order to detect additional bots in a network after one such bot has been discovered. However, none of them consider the provenance in order to propagate the belief. Carter *et al.* [8] propose a PageRank-like link analysis algorithm to compute the threat probability of neighboring nodes in the same graph as the one proposed in [16]. The method improved the effectiveness using threat provenance, yet it does not consider the interrelationship between the risk of hosts and flows. An interesting research issue is to investigate the possibility of applying our method for identifying bots in a network when we have information about one discovered bot.

Several papers investigate how to discover potential dependencies among network flows [17]–[19]. Chen *et al.* [17] introduced the Orion system that discovers dependencies for enterprise applications by using packet headers and timing information. Iliofotou *et al.* [18] proposed the use of Traffic Dispersion Graphs (TDGs) to monitor, analyze, and visualise network traffic by modelling a set of hosts as a social network. Savilla and Ou [19] proposed a PageRank-like algorithm over an attack graph to compute the relative importance of attacker assets. Zand *et al.* [20] proposed an active watermarking approach to detect the dependencies between service and devices. The use of such active models can reduce the number of false positives in our passive causality model. However, it requires one to manipulate the timing and content of the traffic. Zhang *et al.* [21] employed supervised machine learning classifiers to discover triggering relations among network requests. The proposed approach needs training data which is manually labelled. While these approaches exploit techniques for dependency analysis on network activities, our method employs provenance relations among network flows and interdependency between hosts and flows in order to detect high risky hosts and flows. Thus, most of previous approaches are complementary to our and could be used to improve the accuracy of our flow causality model.

There are several approaches which measure the risk based on static network information including hosts vulnerabilities, host impact, and connectivity among them [7], [9]. While they do not consider dynamic [illegible] result of such static risk assessment [illegible] priori knowledge in our approach. [illegible]ledge, no existing work considers th[illegible] dependency between hosts and flo[illegible]twork activities.

VIII. Con[illegible]

This paper has presented a n[illegible]od for hosts and flows which consider the interdependency between the risk of hosts and flows and the flow provenance. Besides proving convergence of our iterative algorithm and providing analytic estimates for its performance, the experimental results over two public datasets show that our method is effective for assigning high risk scores to hosts and flows involved in attacks as well as efficient in terms of processing time for deploying in a high throughput network. As future work, we plan to extend our approach to propose a distributed risk assessment system.

Appendix A
Flow Correlation

We use flow correlation as an indication of flow dependency in our flow dependency graph. The rationale behind this is that a high correlation between two flows is a promising indication for stepping stone detection [5].

In this paper, we choose a set of static and dynamic flow features for computing the flow correlation which can make our correlation analysis more robust against evasion techniques. Accordingly, we first extract a matrix $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ where $\mathbf{x}_i = [x_i^1\ x_i^2\ \ldots\ x_i^m]^T$, $(1 \leq i \leq n)$ represents the i^{th} m-dimensional feature values extracted for flow f_i. After that, we compute the a Pearson correlation coefficient [6] between each two columns of the matrix to obtain the flow correlations.

TABLE VIII show the list of flow features we have selected in the evaluations of this paper. It is important to note that there are two essentially nominal features in the list: destination port and protocol. In order to transform these features into a numerical form, we replaced their values by their probabilities in the matrix.

TABLE VIII
Selected Network Flow Features

Feature	Feature Description
duration	duration in seconds
dst_port	destination port number
protocol	protocol (tcp / udp)
tpno	total number of packets
pno_ctos	number of packets from client to server
pno_stoc	number of packets from server to client
bps	bytes per second
bps_ctos	bytes per second from client to server
bps_stoc	bytes per second from server to client
pps	packet per second
pps_ctos	packet per second from client to server
pps_stoc	packet per second from server to client
fdps	the size of the first data packet in the flow
tppl	total payload packet length
appl	average of payload packet length
vppl	variance of payload packet length
ppl_ctos	payload packet length from client to server
ppl_stoc	payload packet length from server to client
apiat	average of packet inter-arrival time
vpiat	variance of packet inter-arrival time
tmpiat	third moment of packet inter-arrival time
fmpiat	fourth moment of packet inter-arrival time

Appendix B
Proofs

Proof (Lemma 1): Assuming the opposite, if we have at least one cycle in the graph which is a path $(f_1, f_2, \ldots, f_k)$ such that (f_k, f_1) is also an edge, according to Definition 1 for timing information of flows involved in a flow causality, we would have $t(f_1) < t(f_2) < \ldots < t(f_k) < t(f_1)$, which is a contradiction. □

Proof (Lemma 2): At the initial stage of our algorithm all risk scores of flows, hosts and provenances are set to 1. Assuming that the lemma thesis is true at t^{th} stage of iteration, we prove that it is also true for the values obtained at stage $t+1$. Since

$$\begin{aligned}
&\mathrm{npr}^{(t+1)}(f)\\
&= \frac{\mathrm{pr}^{(t+1)}(f)}{M} = \frac{\sum_{p\in t_f} \mathrm{spr}^{(t+1)}(p)}{M}\\
&= \frac{\sum_{p\in t_f} \sum_{\substack{f'\in p\\ f'\neq f}} \mathrm{fr}^{(t)}(f')\mathrm{fcs}(parent(f', p), f')}{M}\\
&\leq \frac{\sum_{p\in t_f} \sum_{\substack{f'\in p\\ f'\neq f}} 1}{M} = \frac{N_t(f)}{\max\{N_t(f') : f' \in F\}} \leq 1.
\end{aligned}$$

Using the above fact, we now have for the risk of flows

$$\begin{aligned}
\mathrm{fr}^{(t+1)}(f) &= c_f \mathrm{dr}^{(t)}(f) + (1 - c_f)\mathrm{npr}^{(t+1)}(f)\\
&= c_f \frac{\mathrm{hr}^{(t)}(\mathrm{src}(f)) + \mathrm{hr}^{(t)}(\mathrm{dst}(f)) + \mathrm{prior}(f)}{3}\\
&\quad +(1 - c_f)\mathrm{npr}^{(t+1)}(f)\\
&\leq c_f \frac{2 + \mathrm{prior}(f)}{3} + (1 - c_f) \leq 1.
\end{aligned}$$

and, using the above for the risk of hosts,

$$\begin{aligned}
&hr^{(t+1)}(h)\\
&= \frac{c_{in} \sum_{f\in F_{I,h}} \mathrm{fr}^{(t+1)}(f)}{|F_{I,h}|} + \frac{(1 - c_{in}) \sum_{f\in F_{O,h}} \mathrm{fr}^{(t+1)}(f)}{|F_{O,h}|}\\
&\leq \frac{c_{in} \sum_{f\in F_{I,h}} 1}{|F_{I,h}|} + \frac{(1 - c_{in}) \sum_{f\in F_{O,h}} 1}{|F_{O,h}|} = 1.
\end{aligned}$$

□

Proof (Lemma 3): We prove the lemma by mathematical induction. The risk of flow f at stage $t+1$ is given by

$$\begin{aligned}&\mathrm{fr}^{(t+1)}(f)\\&= c_f\frac{\mathrm{hr}^{(t)}(\mathrm{src}(f))+\mathrm{hr}^{(t)}(\mathrm{dst}(f))+\mathrm{prior}^{(t)}(f)}{3}\\&\quad+(1-c_f)\frac{\sum_{p\in t_f}\sum_{\substack{f'\in p\\ f'\neq f}} \mathrm{fr}^{(t)}(f')\ [illegible]\ , f')}{[illegible]}\end{aligned}\qquad(16)$$

Base Case: We first prove th [illegible] flows in the case $t=1$.

$$\begin{aligned}\left|\mathrm{fr}^{(2)}(f)-\mathrm{fr}^{(1)}(f)\right| &\le \frac{c_f}{3}\left|\mathrm{hr}^{(1)}(\mathrm{src}(f))-\mathrm{hr}^{(0)}(\mathrm{src}(f))\right|\\&\quad+\frac{c_f}{3}\left|\mathrm{hr}^{(1)}(\mathrm{dst}(f))-\mathrm{hr}^{(0)}(\mathrm{dst}(f))\right|\\&\quad+\frac{c_f}{3}\left|\mathrm{prior}^{(1)}(f)-\mathrm{prior}^{(0)}(f)\right|\\&\quad+(1-c_f)\left|\mathrm{npr}^{(1)}(f)-\mathrm{npr}^{(0)}(f)\right|\\&\le \frac{c_f}{3}+\frac{c_f}{3}+(1-c_f)=1-\frac{1}{3}c_f\end{aligned}$$

Note that $\mathrm{prior}^{(1)}(f)-\mathrm{prior}^{(0)}(f)=0$ because the prior risk scores of flows remain constant during the risk computation process; also according to Corollary 1, the difference between the risk scores is bounded by 1. Now, we similarly prove the base case for the risk of hosts.

$$\begin{aligned}&\left|\mathrm{hr}^{(2)}(h)-\mathrm{hr}^{(1)}(h)\right|\\&\le\frac{c_{in}\sum_{f\in F_{I,h}}\left|\mathrm{fr}^{(2)}(f)-\mathrm{fr}^{(1)}(f)\right|}{|F_{I,h}|}\\&\quad+\frac{(1-c_{in})\sum_{f\in F_{O,h}}\left|\mathrm{fr}^{(2)}(f)-\mathrm{fr}^{(1)}(f)\right|}{|F_{O,h}|}\\&\le\frac{c_{in}\sum_{f\in F_{I,h}}\left(1-\frac{1}{3}c_f\right)}{|F_{I,h}|}+\frac{(1-c_{in})\sum_{f\in F_{O,h}}\left(1-\frac{1}{3}c_f\right)}{|F_{O,h}|}\\&\le c_{in}\left(1-\frac{1}{3}c_f\right)+(1-c_{in})\left(1-\frac{1}{3}c_f\right)=1-\frac{1}{3}c_f.\end{aligned}$$

Note that we used the fact that $\sum_{f\in F_{I,h}} 1=|F_{I,h}|$.

Inductive Step: We assume the bound to be true for iteration t for every flow f and every host h, thus $\left|\mathrm{fr}^{(t+1)}(f)-\mathrm{fr}^{(t)}(f)\right| \le \left(1-\frac{1}{3}c_f\right)^t$ and $\left|\mathrm{hr}^{(t+1)}(h)-\mathrm{hr}^{(t)}(h)\right| \le \left(1-\frac{1}{3}c_f\right)^t$. We first show that the bound is true for iteration $t+1$ for every flow f as follows:

$$\begin{aligned}&\left|\mathrm{fr}^{(t+2)}(f)-\mathrm{fr}^{(t+1)}(f)\right|\\&\le\frac{c_f}{3}\left|\mathrm{hr}^{(t+1)}(\mathrm{src}(f))-\mathrm{hr}^{(t)}(\mathrm{src}(f))\right|\\&\quad+\frac{c_f}{3}\left|\mathrm{hr}^{(t+1)}(\mathrm{dst}(f))-\mathrm{hr}^{(t)}(\mathrm{dst}(f))\right|\\&\quad+(1-c_f)\frac{\sum_{p\in t_f}\sum_{\substack{f'\in p\\ f'\neq f}}\left|\mathrm{fr}^{(t+1)}(f')-\mathrm{fr}^{(t)}(f')\right|}{M}\\&\le\frac{2\times c_f}{3}\left(1-\frac{c_f}{3}\right)^t+(1-c_f)\frac{N_t(f)\left(1-\frac{c_f}{3}\right)^t}{M}\\&\le\frac{2\times c_f}{3}\left(1-\frac{c_f}{3}\right)^t+(1-c_f)\left(1-\frac{c_f}{3}\right)^t=\left(1-\frac{1}{3}c_f\right)^{t+1}\end{aligned}$$

Again, we have used the fact that $\sum_{p\in t_f}\sum_{\substack{f'\in p\\ f'\neq f}} 1=N_t(f)\le\max\{N_t(f') : f'\in F\}$. Now we similarly show that the bound is true at iteration $t+1$ for every host h:

$$\begin{aligned}&\left|\mathrm{hr}^{(t+2)}(h)-\mathrm{hr}^{(t+1)}(h)\right|\\&\le\frac{c_{in}\sum_{f\in F_{I,h}}\left|\mathrm{fr}^{(t+2)}(f)-\mathrm{fr}^{(t+1)}(f)\right|}{|F_{I,h}|}\\&\quad+\frac{(1-c_{in})\sum_{f\in F_{O,h}}\left|\mathrm{fr}^{(t+2)}(f)-\mathrm{fr}^{(t+1)}(f)\right|}{|F_{O,h}|}\\&\le c_{in}\left(1-\frac{c_f}{3}\right)^{t+1}+(1-c_{in})\left(1-\frac{c_f}{3}\right)^{t+1}\\&=\left(1-\frac{1}{3}c_f\right)^{t+1}\end{aligned}$$

□

Proof (Theorem 1): To show convergence, we will show that for every flow f, sequence $\{\mathrm{fr}^{(t)}(f), n\in\mathbb{N}\}$ and for every host h, sequence $\{\mathrm{hr}^{(t)}(h), n\in\mathbb{N}\}$ are Cauchy sequences. This follows immediately from the bound obtained in Lemma 3; for every flow $f\in F$ and $\forall n,m\in\mathbb{N}$, we have

$$\begin{aligned}&\left|\mathrm{fr}^{(n+m)}(f)-\mathrm{fr}^{(n)}(f)\right|\\&\le\left|\mathrm{fr}^{(n+m)}(f)-\mathrm{fr}^{(n+m-1)}(f)\right|\\&\quad+\cdots+\left|\mathrm{fr}^{(n+1)}(f)-\mathrm{fr}^{(n)}(f)\right|\\&\le\left(1-\frac{1}{3}c_f\right)^{n+m-1}+\ \cdots+\left(1-\frac{1}{3}c_f\right)^n\\&=\left(1-\frac{1}{3}c_f\right)^n\frac{1-\left(1-\frac{1}{3}c_f\right)^m}{1-\left(1-\frac{1}{3}c_f\right)}\\&\le\left(1-\frac{1}{3}c_f\right)^n\frac{1}{1-\left(1-\frac{1}{3}c_f\right)}=\frac{3}{c_f}\left(1-\frac{1}{3}c_f\right)^n.\end{aligned}$$

Similarly, for every host h and $\forall n,m\in\mathbb{N}$, we have

$$\left|\mathrm{hr}^{(n+m)}(h)-\mathrm{hr}^{(n)}(h)\right|\le\frac{3}{c_f}\left(1-\frac{1}{3}c_f\right)^n.$$

Thus, when n is sufficiently large, for all m the values of $\left|\mathrm{fr}^{(n+m)}(f)-\mathrm{fr}^{(n)}(f)\right|$ and $\left|\mathrm{hr}^{(n+m)}(h)-\mathrm{hr}^{(n)}(h)\right|$ can be made arbitrarily small. The claim now follows from the fact that all Cauchy sequences on reals are convergent. □

Proof (Lemma 4): Since by Theorem 1

$$\left|\mathrm{fr}^{(n+m)}(f)-\mathrm{fr}^{(n)}(f)\right|\le\frac{3}{c_f}\left(1-\frac{1}{3}c_f\right)^n;$$

$$\left|\mathrm{hr}^{(n+m)}(h)-\mathrm{hr}^{(n)}(h)\right|\le\frac{3}{c_f}\left(1-\frac{1}{3}c_f\right)^n,$$

the same inequalities hold for the limit as $m\to\infty$; thus, denoting the limits by $\mathrm{fr}^{(\infty)}(f)$ and $\mathrm{hr}^{(\infty)}(f)$ respectively, we obtain

$$\left|\mathrm{fr}^{(\infty)}(f)-\mathrm{fr}^{(n)}(f)\right| \le \frac{3}{c_f}\left(1-\frac{1}{3}c_f\right)^n;$$
$$\left|\mathrm{hr}^{(\infty)}(h)-\mathrm{hr}^{(n)}(h)\right| \le \frac{3}{c_f}\left(1-\frac{1}{3}c_f\right)^n.$$

Thus, both $\left|\mathrm{fr}^{(\infty)}(f)-\mathrm{f}$ [illegible] and $\left|\mathrm{hr}^{(\infty)}(h)-\mathrm{hr}^{(n)}(h)\right| \le \varepsilon$ w[illegible] $< \varepsilon$; taking the logarithm of both s[illegible]ppens just in case $\log_{(1-\frac{1}{3}c_f)}\frac{3}{c_f}+n$ [illegible]proves our claim with $\delta=\log_{(1-\frac{1}{3}c_f)}$ [illegible] □

Proof (Theorem 2): We pr[illegible] fixed point in our iterative algorithm [illegible]tion". If the algorithm does not provi[illegible]en we have at least two different fixed points which provide different risk scores for both hosts and flows. Assume the provided risk score for flow f (host h) by the first and second fixed points are $\mathrm{fr}_1^*(f)$ and $\mathrm{fr}_2^*(f)$ ($\mathrm{hr}_1^*(h)$ and $\mathrm{hr}_2^*(h)$) respectively. Denote the corresponding normalized flow provenance risk score by $\mathrm{npr}_k^*(f)$ for the k-th fixed point. We first prove the following inequalities by mathematical induction.

$$\forall n\in\mathbb{N},\quad \left|\mathrm{fr}_1^*(f)-\mathrm{fr}_2^*(f)\right| \le \left(1-\frac{1}{3}c_f\right)^n. \qquad (17)$$
$$\forall n\in\mathbb{N},\quad \left|\mathrm{hr}_1^*(h)-\mathrm{hr}_2^*(h)\right| \le \left(1-\frac{1}{3}c_f\right)^n. \qquad (18)$$

Base Case: From equation (16) in the fixed point and for the case $n=1$, we have

$$\begin{aligned}&\left|\mathrm{fr}_1^*(f)-\mathrm{fr}_2^*(f)\right|\\ &=\Bigg|(1-c_f)(\mathrm{npr}_1^*(f)-\mathrm{npr}_2^*(f))\\ &\quad+c_f\frac{\mathrm{hr}_1^*(\mathrm{src}(f))-\mathrm{hr}_2^*(\mathrm{src}(f))+\mathrm{hr}_1^*(\mathrm{dst}(f))-\mathrm{hr}_2^*(\mathrm{dst}(f))}{3}\Bigg|\\ &\le\frac{c_f}{3}\left|\mathrm{hr}_1^*(\mathrm{src}(f))-\mathrm{hr}_2^*(\mathrm{src}(f))\right|\\ &\quad+\frac{c_f}{3}\left|\mathrm{hr}_1^*(\mathrm{dst}(f))-\mathrm{hr}_2^*(\mathrm{dst}(f))\right|\\ &\quad+(1-c_f)\left|\mathrm{npr}_1^*(f)-\mathrm{npr}_2^*(f)\right|\\ &\le\frac{c_f}{3}+\frac{c_f}{3}+(1-c_f)=1-\frac{1}{3}c_f\end{aligned}$$

Now, we similarly prove the base case ($n=1$) for the risk of hosts.

$$\begin{aligned}&\left|\mathrm{hr}_1^*(h)-\mathrm{hr}_2^*(h)\right|\\ &\le\frac{c_{in}\sum_{f\in F_{I,h}}\left|\mathrm{fr}_1^*(f)-\mathrm{fr}_2^*(f)\right|}{\left|F_{I,h}\right|}\\ &\quad+\frac{(1-c_{in})\sum_{f\in F_{O,h}}\left|\mathrm{fr}_1^*(f)-\mathrm{fr}_2^*(f)\right|}{\left|F_{O,h}\right|}\\ &\le\frac{c_{in}\sum_{f\in F_{I,h}}(1-\frac{1}{3}c_f)}{\left|F_{I,h}\right|}+\frac{(1-c_{in})\sum_{f\in F_{O,h}}(1-\frac{1}{3}c_f)}{\left|F_{O,h}\right|}\\ &\le c_{in}\left(1-\frac{1}{3}c_f\right)+(1-c_{in})\left(1-\frac{1}{3}c_f\right)=1-\frac{1}{3}c_f.\end{aligned}$$

Note that we used the fact that $\sum_{f\in F_{I,h}}1=\left|F_{I,h}\right|$.

Inductive Step: We assume the (17) and (18) to be true in the case n for every flow f and every host h, thus $\left|\mathrm{fr}_1^*(f)-\mathrm{fr}_2^*(f)\right| \le (1-\frac{1}{3}c_f)^n$ and $\left|\mathrm{hr}_1^*(h)-\mathrm{hr}_2^*(h)\right| \le (1-\frac{1}{3}c_f)^n$. We first show that inequality (17) is true in the case $n+1$ for every flow f as:

$$\begin{aligned}&\left|\mathrm{fr}_1^*(f)-\mathrm{fr}_2^*(f)\right|\\ &\le\frac{c_f}{3}\left|\mathrm{hr}_1^*(\mathrm{src}(f))-\mathrm{hr}_2^*(\mathrm{src}(f))\right|\\ &\quad+\frac{c_f}{3}\left|\mathrm{hr}_1^*(\mathrm{dst}(f))-\mathrm{hr}_2^*(\mathrm{dst}(f))\right|\\ &\quad+(1-c_f)\left|\mathrm{npr}_1^*(f)-\mathrm{npr}_2^*(f)\right|\\ &\le\frac{2\times c_f}{3}\left(1-\frac{1}{3}c_f\right)^n+(1-c_f)\frac{N_t(f)\left(1-\frac{1}{3}c_f\right)^n}{M}\\ &\le\frac{2\times c_f}{3}\left(1-\frac{1}{3}c_f\right)^n+(1-c_f)\left(1-\frac{1}{3}c_f\right)^n\\ &=\left(1-\frac{1}{3}c_f\right)^{n+1}\end{aligned}$$

Again, we have used the fact that $\sum_{p\in t_f}\sum_{\substack{f'\in p\\ f'\neq f}}1=N_t(f)\le\max\{N_t(f'):f'\in F\}$. Now we similarly show that the bound is true for the iteration $n+1$ for every host h as follows:

$$\begin{aligned}&\left|\mathrm{hr}_1^*(h)-\mathrm{hr}_2^*(h)\right|\\ &\le c_{in}\left(1-\frac{1}{3}c_f\right)^{n+1}+(1-c_{in})\left(1-\frac{1}{3}c_f\right)^{n+1}\\ &=\left(1-\frac{1}{3}c_f\right)^{n+1}\end{aligned}$$

Now, we proved the inequalities hold for any $n\in\mathbb{N}$; thus, we obtain

$$\left|\mathrm{fr}_1^*(f)-\mathrm{fr}_2^*(f)\right| \le \lim_{n\to\infty}\left(1-\frac{1}{3}c_f\right)^n=0;$$
$$\left|\mathrm{hr}_1^*(h)-\mathrm{hr}_2^*(h)\right| \le \lim_{n\to\infty}\left(1-\frac{1}{3}c_f\right)^n=0.$$

The above inequalities are possible if and only if $\mathrm{fr}_1^*(f)=\mathrm{fr}_2^*(f)$ and $\mathrm{hr}_1^*(h)=\mathrm{hr}_2^*(h)$, which proves our claim. □

References

[1] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton, NJ, USA: Princeton Univ. Press, Feb. 2012.

[2] S. Wang, R. State, M. Ourdane, and T. Engel, "RiskRank: Security risk ranking for IP flow records," in *Proc. CNSM*, Oct. 2010, pp. 56–63.

[3] M. Rezvani, A. Ignjatovic, E. Bertino, and S. Jha, "Provenance-aware security risk analysis for hosts and network flows," in *Proc. NOMS*, May 2014, pp. 1–8.

[4] W. T. Strayer, C. Jones, B. Schwartz, S. Edwards, W. Milliken, and A. Jackson, "Efficient multi-dimensional flow correlation," in *Proc. LCN*, Oct. 2007, pp. 531–538.

[5] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*. New York, NY, USA: Springer-Verlag, 2010.

[6] H.-S. Lim, Y.-S. Moon, and E. Bertino, "Provenance-based trustworthiness assessment in sensor networks," in *Proc. DMSN*, 2010, pp. 2–7.

[7] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 1, pp. 61–74, Jan./Feb. 2012.

[8] K. M. Carter, N. Idika, and W. W. Streilein, "Probabilistic threat propagation for network security," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 9, pp. 1394–1405, Sep. 2014.

[9] M. A. Rahman and E. Al-Shaer, "A formal approach for network security management based on qualitative risk analysis," in *Proc. IM*, May 2013, pp. 244–251.

[10] N. Feng and M. Li, "An information systems security risk assessment model under uncertain environment," *Appl. Soft Comput.*, vol. 11, no. 7, pp. 4332–4340, Oct. 2011.

[11] *Scan 18 and Challenge 1 of the Forensic Challenge 2010*. [Online]. Available: http://www.honeynet.org/challenges, accessed Aug. 1, 2013.

[12] MIT Lincoln Laboratory. (2014). [illegible] *DARPA Intrusion Detection, Lincoln Lab*[illegible] [Online]. Available: http://www.ll.mit.edu/[illegible]

[13] *Emsisoft Portlist—All Know*[illegible]*ts of Malware, Trojans, Spyware* [illegible]ailable: http://www.emsisoft.com/en/kb/p[illegible]

[14] Google. (2014). *Google Safe* [illegible]ailable: http://developers.google.com/safe[illegible]

[15] A. Houmansadr and N. Borisov, [illegible] to link correlated network flows," in *Pri*[illegible] Lecture Notes in Computer Science). Be[illegible], 2013, pp. 205–224.

[16] B. Coskun, S. Dietrich, and N. Memon, "Friends of an enemy: Identifying local members of peer-to-peer botnets using mutual contacts," in *Proc. ACSAC*, 2010, pp. 131–140.

[17] X. Chen, M. Zhang, Z. M. Mao, and P. Bahl, "Automating network application dependency discovery: Experiences, limitations, and new solutions," in *Proc. USENIX OSDI*, 2008, pp. 117–130.

[18] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese, "Network monitoring using traffic dispersion graphs (TDGs)," in *Proc. IMC*, 2007, pp. 315–320.

[19] R. E. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *Proc. ESORICS*, 2008, pp. 18–34.

[20] A. Zand, G. Vigna, R. Kemmerer, and C. Kruegel, "Rippler: Delay injection for service dependency detection," in *Proc. IEEE INFOCOM*, Apr./May 2014, pp. 2157–2165.

[21] H. Zhang, D. D. Yao, and N. Ramakrishnan, "Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery," in *Proc. ASIA CCS*, 2014, pp. 39–50.

Mohsen Rezvani (S'12) received the B.Sc. degree in computer engineering from the Amirkabir University of Technology, and the M.Sc. degree in computer engineering from the Sharif University of Technology, Iran. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, University of New South Wales, Australia. His research focuses on trust and reputation systems.

Verica Sekulic received the bachelor's degree in mathematics from the University of Montenegro, Montenegro. She is currently pursuing the master's degree with the School of Computer Science and Engineering, University of New South Wales,, Australia.

Aleksandar Ignjatovic received the Ph.D. degree in mathematical logic from the University of California at Berkeley. He is currently an Associate Professor with the School of Computer Science and Engineering, University of New South Wales, Australia. His current research interests include approximation theory, sampling theory and applied harmonic analysis, algorithm design, and applications of mathematical logic to computational complexity theory.

Elisa Bertino (F'02) is currently a Professor of Computer Science with Purdue University and serves as Research Director of the Center for Education and Research in Information Assurance and Security. Her main research interests include security, privacy, digital identity management systems, database systems, distributed systems, and multimedia systems. She is a fellow of the Association for Computing Machinery, and has been named a Golden Core Member for her service to the IEEE Computer Society.

Sanjay Jha (SM'08) received the Ph.D. degree in computer science from the University of Technology, Sydney, Australia. He is currently a Professor of Computer Science with University of New South Wales, Australia. His research activities cover a wide range of topics in networking, including wireless sensor networks, adhoc/community wireless networks, resilience and multicasting in IP networks, and security protocols for wired/wireless networks.