Iterative Security Risk Analysis for Network Flows Based on Provenance and Interdependency

Mohsen Rezvani Aleksandar Ignjatovic Sanjay Jha

University of New South Wales, Australia
{mrezvani,ignjat,sanjay}@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-201305
February 2013

Abstract

Discovering high risk network flows and hosts in a high throughput network is a challenging task of network monitoring. Emerging complicated attack scenarios such as DDoS attacks increases the complexity of tracking malicious and high risk network activities within a huge number of monitored network flows. To address this problem, we propose an iterative framework to assessing risk scores for hosts and network flows. To obtain risk scores of flows, we take into account two properties, flow attributes and flow provenance. Also, our iterative risk assessment measures the risk scores of hosts and flows based on an interdependency property where the risk score of a flow influences the risk of its source and destination hosts, and the risk score of a host is evaluated by risk scores of flows initiated by or terminated at the host. Moreover, the update mechanism in our framework allows flows to keep streaming into the system while our risk assessment method performs an online monitoring task. The experimental results show that our approach is effective in detecting high risk hosts and flows as well as sufficiently efficient to be deployed in high throughput networks compared to other algorithms.

1 Introduction

A significant challenge for monitoring of large enterprise networks is the complexity of extracting risky network flows from the tremendous quantity of flow records. However, such management task aids security management by identifying the most likely malicious activities thus helping take effective countermeasures. A security relevant flow can be defined recursively as either an IP flow which transports malicious content or which has initiated another security relevant flow [20]. Such recursively defined security relevant IP flow makes its detection rather complicated. Moreover, in order to provide a ranking of the risk level of a network flow record, one needs to design a comprehensive framework which takes into account various parameters that may affect the level of risk.

Distributed attack scenarios such as DDoS attacks and Botnet initiated attacks are examples of attacks which generate security relevant flows satisfying the above recursive definition. One of the best illustrations of the inter-flow relationship as specified in the recursive definition of a security relevant flow appears when an attacker creates a web session on a public web server hosted within DMZ, compromising the web server. While the web related flow traffic to such server is a permitted flow in the security policy, the attacker can initiate a new connection from the compromised server to a another server in protected network zone which allows it to do rootkit downloading, massive SPAM sending or port scanning on the second server, where such activities are not allowed. This illustrates the fact that, in order to adequately evaluate the level of risk of the initial flow of this scenario, we need to consider whole interdependency risk relationship among recorded network flows.

It is clear that in order to address the problem of risky flow monitoring, we need a comprehensive solution which considers whole interdependency risk relationship among the recorded network flows as well as the hosts initiating and targeted by the flows. In the recent research, the idea of employing link analysis techniques such as PageRank [2] and HITS [10] has been proposed for detecting relevant IP flows by [20, 19], but they analyzed levels of risk of hosts and levels of risk of flows separately, without considering their interdependency. Moreover, this separately risk assessment leads to exploit two different dependency graphs for measuring the level of risk for hosts and flows which highly decrease the efficiency of these methods for high throughput networks. In this paper, we propose an interdependency risk model for ranking the riskiness of network flows as well as the related hosts. The idea for our proposal is inspired by the provenance-based trust model proposed in [5, 11], albeit it used in the context of sensor networks and data management.

In our 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 most of its related flows are risky. With such interdependency in mind, we develop an iterative algorithm for calculating the level of risk of hosts as well as the level of risk of network flows. Moreover, we take into account two different aspects that may influence the level of risk of a network flow, the risk of flow attributes and the risk of flow provenance. The former is defined by an aggregation of the risk level of the source and destination hosts of a flow and a predefined risk level of service of the network flow. The latter concept of flow provenance is specified based on the recursive definition of security relevant flow and we will formally define flow provenance based on all other flow records which can be considered as a consequence of such a flow. In addition, risk scores for hosts and flows are measured in sliding time windows, which is an important feature allowing our algorithm to be used in real time.

We summarize our contributions as follows:

- A significant challenge of handling a flow dependency graph is the scalability, due to possibly huge number of flows in the graph for high throughput networks. Therefore, we propose a scalable algorithm for risk assessment based on flow provenance.
- We formulate the concept of flow provenance to measure the level of risk of network flows, based on a recursive relationship between network flow records.
- In order to facilitate monitoring of malicious network activities, we propose a risk assessment model based on an interdependency relationship between the level of risk for hosts and network flows.
- We develop an iterative algorithm to simultaneously compute the level of risk for both hosts and flows. Our experimental results demonstrate its efficiency.

The reminder of the paper is organized as follows. The assumptions and preliminary definitions are specified in Section 2. Section 3 presents the details of our proposed risk computation model. Performance analysis and experimental results are presented in Section 4. The related work is presented in Section 5. Concluding remarks are made in Section 6.

2 Preliminary Definitions

In this section, we describe several concepts used in our risk computation model. These concepts include a basic definition of NetFlow, flow dependency graph and risky path based on the graph. Throughout this report, the terms IP flow and NetFlow are used interchangeably.

2.1 Network Flow

Handling full packet capture in a large enterprise network for security analysis is a significant challenge in terms of time and space complexity. Network flow can be defined as a unidirectional or bidirectional sequence of IP packets or frames that have a few common attributes such as being a member of the same TCP connection or UDP session. By aggregating the packets and frames of a flow, a summary information about the flow can be extracted and applied for network management in order to mitigate the issue of tremendous quantity of network packets in large networks.

Figure 2.1 illustrates the overall framework of NetFlow proposed by Cisco Systems [21]. Three main components in this framework consist of NetFlow Explorer, NetFlow Collector and Analysis Console. The network devices such as switches and routers can support the role of NetFlow explorer by sending the flow packets to the collectors. The NetFlow collector can receive the flow packets

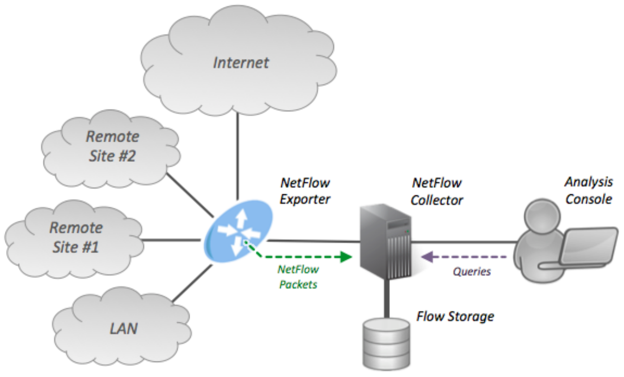


Figure 2.1: NetFlow architecture [21].

Table 2.1: NetFlow samples based on session 5-tuple.

id	start_t	duration	src_ip	src_port	dest_ip	dest_port	protocol
1	952438901	5	172.16.112.50	33354	172.16.114.50	80	tcp
2	952438905	219	172.16.113.168	25314	172.16.112.100	21	tcp
3	952438916	10	172.16.118.255	520	172.16.112.20	520	udp
4	952438920	30	194.27.251.21	43740	172.16.112.194	25	tcp
5	952438922	3	172.16.112.100	1439	172.16.115.20	53	udp

from several explorers and supports a query language on its flow storage for different flow analyzers. In this research, the flow analysis consists of measuring the risk score of flows based on an iterative computational framework.

As an example of NetFlow, a part of flows collected from the public packet captured by DARPA project [6] is shown in Table 2.1. This table illustrates only a summary of information for bidirectional and session based flows including start time, duration and a 5-tuple of the TCP/UDP session, while our implemented algorithm (discussed in Section 4) can extract a few more attributes for each session.

2.2 Flow Dependency Graph

The main idea behind the flow dependency graph is modeling causality among network flow records in order to measure the level of risk of a flow, based on its likelihood to be the root cause of attacks in the network. We will employ the flow dependency graph in order to extract the probable attack scenarios which a network flow has initiated. This dependency graph is based on the proposed flow dependency in [19].

The significant benefit of flow dependency graph in our risk assessment model is that it allows to represent the casuality between network flows. As defined in [19], a flow record A causes a flow record B, if flow A triggered flow B. In other words, in a particular time window, the flow record B arrives after the flow A and the source address of B is same as the destination address of flow A. This causality can be considered as resulting from a possible attack scenario

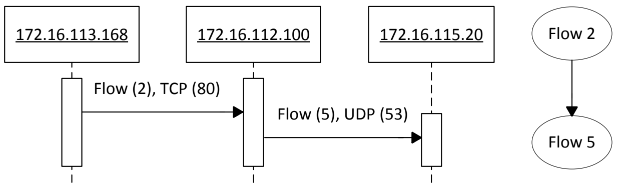


Figure 2.2: An example for flow dependency graph.

in which the attacker creates the flow A in the first step (from host 1 to host 2) and after compromising the destination of this flow (host 2), she establishes the second flow B (from host 2 to host 3).

Figure 2.2 shows an example of a flow dependency graph for two dependent network flows from Table 2.1. The flow dependency graph is a directed graph based on a sequence of flow records monitored during a particular time window. Nodes are flows in the sequence, while the edges represent possible causality between nodes. In other words, we create an edge from node A to node B in the dependency graph, if the flow A causes flow B based on the above causality definition.

A significant challenge in maintaining a flow dependency graph is the scalability, due to possibly huge number of flows in the graph for high throughput networks. Therefore, we propose a scalable algorithm for measuring the level of risk from this graph in Section 3.

2.3 Risky Paths

We define the risky path in a flow dependency graph, to be used to represent a potential attack scenario initiated by a flow record.

Definition 1. ***Risky Path.*** *A risky path of a flow record f is every simple path in the flow dependency graph starting at node f.*

In the risk evaluation algorithm, we will consider the risk level of risky paths for each flow record. It is clear that there may be more than one risky path for a flow record in the flow dependency graph. We can consider either the highest risk value or sum of risk values of all risky paths for a flow record; in our implementations we opted for the highest risk path. More precisely, the flows which participated in more risky paths will be assigned a higher risk score. We now define a new graph, called the flow provenance, whose purpose is to simplify evaluation of level of risk for a flow record.

Definition 2. ***Flow provenance.*** *The flow provenance t_f of a network flow f is a directed graph with the following properties: (1) t_f is a subgraph of the flow dependency graph; (2) t_f includes node f and all nodes which are accessible from node f by a risky path; (3) t_f contains all the edges of the flow dependency graph that are between two nodes in t_f.*

3 Provenance-Aware Risk Computation

We extend the common 5-tuple unique flow identifiers with two additional attributes, the start time and the duration of the flow, in order to construct the flow dependency graph. Moreover, we will define several different risk scores in

Table 3.1: Notation used in this paper.

F	the set of all flows in the current time window
$\mathrm{hr}(h)$	the risk score of a host h
$\mathrm{ar}(f)$	the risk score of a combination of attributes of a flow f
$\mathrm{sr}(s)$	the risk score of a network service s
$\mathrm{pr}(f)$	the risk score of a provenance of a flow f
$\mathrm{fr}(f)$	the risk score of a flow f
$\mathrm{spr}(p)$	the risk score of a simple path p in the flow dependency graph

our risk computational model. Table 3.1 contains a summary of notations used in this paper.

3.1 Iterative Framework

The main idea behind our risk computation system is to model the network flow monitoring activity as a data management problem. Therefore, the risk assessment can be modeled as a negative trust (distrust) computation. Authors in [5, 11] proposed a provenance-based model for data providers with assuming an interdependency relationship between data items and data providers. We model network flows as data items which are provided by network hosts. Hence, we define an interdependency relationship between network hosts and flow records in order to formulate the risk measurement for both of them.

The provenance concept in data trustworthiness models presents the path of provisioning a data item, whereas we have defined the concept of flow provenance for a flow record based on its probable consequences for generating further flows. While the data provenance concept is related to the process of generating the data item by various data providers, the notion of flow provenance which we have introduced represents network activities which are generated by the flow record. Note that in the usual sense of the provenance as used in data management, the path of generating data by various data providers is used as a parameter for measuring data trustworthiness. In our definition, the direction is somewhat reverse, in the sense that the risk of a flow is impacted by riskiness of the further flows which are caused by such a flow.

In our risk computation model, the risk scores are assigned to both hosts and flows, in an interdependent manner. Accordingly, the risk of a host is computed by aggregation of the risk scores of network flows related to the host. Furthermore, the level of risk of a flow record is partially measured by the risk scores of source and destination hosts and service type of the flow. Figure 3.1 shows this interdependency between the host and the flow risk levels.

In order to deploy our risk assessment system on an ISP network, we need to provide a way of handling a large number of flow records. The monitored flows are an input stream for our system and they are handled in overlapping time windows. In the other words, we apply our risk assessment method on the current window, also using as an initial risk evaluation value for hosts and flows obtained from the previous window. Thus, risk evaluations in each subsequent window are obtained via an updating mechanism from the corresponding values from the previous window.

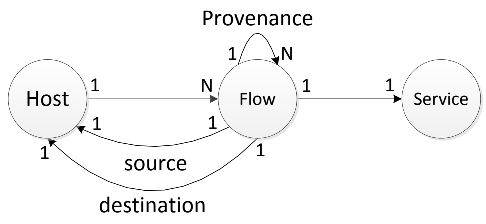
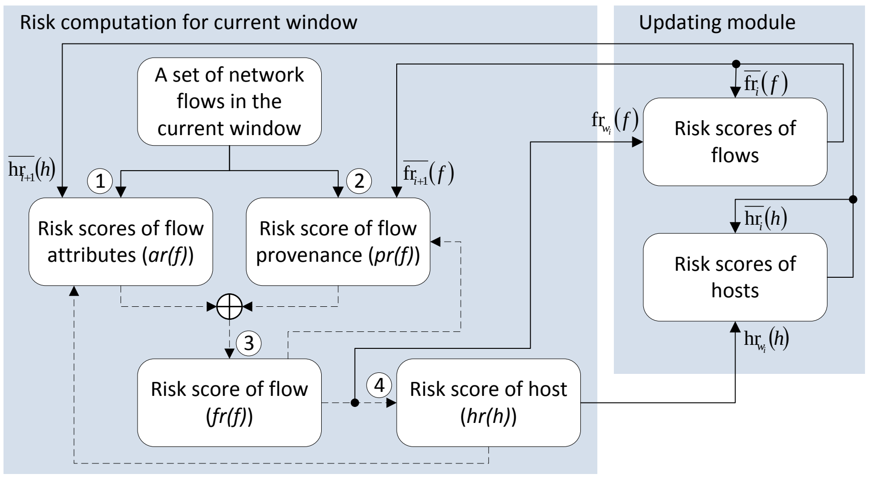


Figure 3.1: Interdependency relationship between host and flow.



Risk computation for current window
Updating module
A set of network flows in the current window
$\overline{\mathrm{hr}_{i+1}}(h)$
1
2
$\overline{\mathrm{fr}_{i+1}}(f)$
$\overline{\mathrm{fr}_i}(f)$
$\mathrm{fr}_{w_i}(f)$
Risk scores of flows
$\overline{\mathrm{hr}_i}(h)$
Risk scores of flow attributes (ar(f))
Risk score of flow provenance (pr(f))
Risk scores of hosts
3
$\mathrm{hr}_{w_i}(h)$
4
Risk score of flow (fr(f))
Risk score of host (hr(h))

Figure 3.2: An iterative framework of computing risk scores of hosts and flows.

Figure 3.2 illustrates the iterative framework for computing risk scores of hosts and flows. We first explain the overall architecture of our system, explaining the details in the subsequent sections. As shown in this figure, two main modules of our framework are risk evaluation component for current time window and the updating mechanism. Dashed lines are traversed in each iteration within a computation for each window; the solid lines are traversed from one window to the next one.

For a set of monitored flow records in the current window, we iteratively compute the risk scores for flow attributes and flow provenance respectively. In each iteration these two computed risk scores for each flow are combined together in order to obtain the new value of the risk score the of the flow; such a score is used in the subsequent iteration to update the risk score of provenance as well as to compute the risk scores of the hosts; the risk score of the host is then also used in the next iteration to obtain the risk scores of the attributes.

Such iterative risk computations for the current window will be repeated until the changes in the risk scores become negligible. After finishing such iterative risk computation for the current time window, the risk scores of hosts and the risk scores of the flows are passed to the updating module. In the updating module, the current results will be combined with the results from the previous time window by producing a weighted sum of such values and then a computation for next time window will be started. We explain the details of such computation process of this framework in the next sections.

3.2 Risk Score Computation for Network Flows

The risk score of each network flow is obtained by aggregation of the risk score of its attributes and the risk score of its provenance; the computation of the risk score of the provenance exploits the flow dependency graph.

Risk of Flow Attributes

The risk score of a flow attributes is calculated based on the risk scores of source host, destination host and the service type of the flow. For the risk computation of source and destination hosts, we will use the current risk score of related hosts, while for computing the risk of flow service, we need a prior knowledge about the risky services in the network services (see ① in Figure 3.2).

Although we can roughly assign a risk level to the requested service in each flow, we allowed the possibility that the risk level for network services is supplied by the administrator, based on his prior knowledge about the network services. Emsisoft Portlist [7] define the well know TCP and UDP ports used for malware, trojans, spyware and viruses. Based on this port list, a service risk level for each flow is assigned. In other words, if the destination port of a flow record is in this port list, we accordingly assign a higher service risk for such a flow.

Having obtained the risk scores of all three attributes of a network flow, we can define the risk score of attributes of flow f as a simple sum as follows[1]:

$$\mathrm{ar}(f) = \mathrm{hr}(\mathrm{src}(f)) + \mathrm{hr}(\mathrm{dst}(f)) + \mathrm{sr}(\mathrm{srv}(f)), \tag{3.1}$$

where src(f), dst(f) and srv(f) denote the source, destination and destination port number of flow f, respectively. Function sr(s) in above equation represents the risk score for a flow service defined as follows:

$$\mathrm{sr}(s) = \begin{cases} 1 & s \in \texttt{Emsisoft Portlist} \\ 0 & \texttt{Otherwise} \end{cases}$$

Risk of Flow Provenance

The risk score of a flow provenance is obtained from the risk scores of nodes of the provenance (see ② in Figure 3.2). As defined in Section 2, a flow provenance can contain a number of flows and risky paths. For simplicity, we first assume that the flow provenance is a graph without any cycles. We will later show that that our iterative algorithm for computing risk scores also handles flow provenances with cycles correctly.

Figure 3.3a illustrates a simple example of provenance for flow $f1$. In this example, there are three different risky paths for this flow record: $f1 \rightarrow f2 \rightarrow f3$, $f1 \rightarrow f4 \rightarrow f5 \rightarrow f3$, and $f1 \rightarrow f4 \rightarrow f6 \rightarrow f5 \rightarrow f3$.

Once a flow provenance is represented by a number of risky paths, we can compute the risk score of a flow provenance as the maximum risk score of its risky paths [2]. Moreover, the risk score of a risky path is obtained as a sum of

[1]All of the functions in the equations below can be found in Table 3.1.

[2]We can also use the sum of the risk scores of risky flows for computing the risk score of a flow provenance. In this paper we evaluate the performance of our method using the maximum of risk scores; in future version of this paper we will also evaluate our method with using the sum of the risk scores. Choosing the sum has an advantage of making all operations linear, which facilitates the proof of convergence of our method, to be presented in the forthcoming extension of this work.

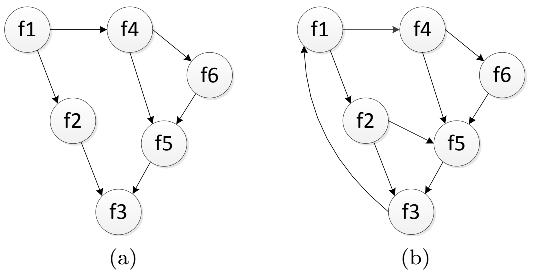


Figure 3.3: Simple and cyclic risky consequences for flow $f1$.

the risk scores of flows within the path:

$$\text{spr}(p) = \sum_{f \in p} \text{rf}(f) \tag{3.2}$$

$$\text{pr}(f) = \underset{p \in t_f}{\text{argmax}}\{\text{pr}(p)\} \tag{3.3}$$

where spr(p) and pr(f) denote the risk scores of a risky path p and a flow provenance of flow f, respectively.

In order to calculate the risk score of provenances of monitored flows based on Equations 3.2 and 3.3, we propose a customized DFS (Depth First Search) algorithm on the flow dependency graph. Algorithm 1 shows DFS computation of the risk scores of provenance for all flows in the graph. Because the flow graph dependency is mostly a sparse graph, we insert a new virtual node in the graph with edges to all existing nodes, in order to allow the DFS algorithm to traverse all nodes of the graph; this new virtual node is the starting node for DFS algorithm. The algorithm recursively computes the risk score of provenance for each flow (represented as a node in the graph). As shown in Algorithm 1, the risk scores of flow provenances are stored in a global array *pr*.

Algorithm 1 Recursive algorithm for computing risk of flow provenance

```
1:  procedure ProvenanceRisk(Graph $G$, Vertex $v$)
2:      if not visited[$v$] then
3:          $visited[v] \leftarrow true$
4:          Let $f_1, \ldots, f_k$ be $k$ child nodes of $v$
5:          for all $i$ $(i = 1, \ldots, k)$ do
6:              if not visited[$f_i$] then
7:                  ProvenanceRisk($G$, $f_i$)
8:              end if
9:          end for
10:         $\text{pr}(v) \leftarrow \underset{i \in \{1,\ldots,k\}}{\text{argmax}}\{\text{fr}(f_i) + \text{pr}(f_i)\}$
11:     end if
12: end procedure
```

A cyclic flow dependency graph is a dependency graph containing at least one cycle (Figure 3.3b). It is obvious that our DFS searching solution proposed in Algorithm 1 prevents any infinite loop in a cyclic graph by labeling the nodes as *visited*, although it may influence the risk computation for some flow provenances. The proposed iterative process resolves the problem of computing risk score of flow provenance for cyclic flow dependency graphs. Although some flow provenances are assigned incorrect risk scores in the initial round of iteration, the next round computes the correct risk scores for them.

Flow Risk Aggregation

As we described, the risk score of a flow is computed by aggregation of the risk values of attributes and provenance of the network flow (see ③ in Figure 3.2). We apply a simple aggregation by a weighted summation of these two risk values in order to compute the total risk score of each flow:

$$\hat{\mathrm{fr}}(f) = c_f \mathrm{ar}(f) + (1 - c_f)\mathrm{pr}(f) \tag{3.4}$$

where F is the set of all monitored flows in the current time window and c_f is a constant, $0 \leq c_f \leq 1$, which represents the impact of risk score of flow attributes in the computing the total risk of the flow[3]. In other words, an administrator can select a large value for c_f, when the risky ports as a part of flow attributes are well defined for the network. On the other hand, for prioritizing the risk of complicated attack scenarios, the administrator can assign a small value for this constant and consider more impacts for flow provenance.

3.3 Risk Score for Hosts

The risk score of a host is computed based on its engagement in risky network activities (see ④ in Figure 3.2). The network activities of a host are specified by its incoming and outgoing network flows within the current time window. Moreover, there is a significant difference for impacts of incoming and outgoing risks which needs to be taken into account in the computation process. Hence, we propose the following equation, used in our experimental evaluations:

$$\hat{\mathrm{hr}}(h) = \frac{c_{in} \sum\limits_{f \in F_{I,h}} \mathrm{fr}(f)}{|F_{I,h}|} + \frac{(1 - c_{in}) \sum\limits_{f \in F_{O,h}} \mathrm{fr}(f)}{|F_{O,h}|} \tag{3.5}$$

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

In equation 3.5, c_{in} is a constant of $0 \leq c_{in} \leq 1$ chosen to reflect the impact of incoming flows in computing the risk score. The incoming flows should have a higher impact on the risk value, and we allow such a constant to be adjusted by the network administrator[4].

3.4 Iterative Algorithm

As we explained, an iterative algorithm is employed for computing the risk scores for flows and hosts within each time window. Algorithm 2 shows such iterative process; two host and flow risk vectors (respectively **hr** and **fr**) are inputs from

[3]In the experiment we set $c_f = 0.5$ to equally reflect the importance of $ar(f)$ and $pr(f)$. Future work will study the impact of selecting different values of this parameter.

[4]In our experiments we set $c_{in} = 0.75$ giving more importance to incoming flows than the outgoing ones in computations of risk scores of hosts.

the previous time window and an input vector **F** is a set of monitored flow records for current time window.

Algorithm 2 Iterative algorithm for risk computation within each time window.

```
1: procedure RiskComputation($\mathbf{hr}$, $\mathbf{fr}$, $\mathbf{F}$)
2:     repeat
3:         Compute $\mathrm{af}(f)$ for all $f \in \mathbf{F}$ using equation (3.1)
4:         Compute $\mathrm{pr}(f)$ for all $f \in \mathbf{F}$ using equation (3.3)
5:         Compute $\hat{\mathrm{fr}}(f)$ for all $f \in \mathbf{F}$ using equation (3.4)
6:         Compute $\hat{\mathrm{hr}}(h)$ for all host $h$ involved in $\mathbf{F}$ using equation (3.5)
7:         Normalize $\hat{\mathbf{fr}}$ and $\hat{\mathbf{hr}}$
8:         $\mathbf{fr} \leftarrow \hat{\mathbf{fr}}$
9:         $\mathbf{hr} \leftarrow \hat{\mathbf{hr}}$
10:    until the change of risk values is smaller than the threshold value
11:    Return $\mathbf{fr}$ and $\mathbf{hr}$
12: end procedure
```

3.5 Updating Process

As represented on Figure 3.2, the updating mechanism is one of the two main modules of our risk computation architecture. This module allows the flow records to keep streaming into the risk computation system. In the updating process and before starting a risk computation for the next time window w_{i+1}, we first obtain initial risk scores $\bar{\mathrm{hr}}_{i+1}(h)$ of each host h, to be used as initial values for computation for the window w_{i+1}, to be provided by the updating mechanism, as a weighted sum of the corresponding values $\mathrm{hr}_{w_i}(h)$ from the previous window w_i and $\bar{\mathrm{hr}}_i(h)$ from the previous values obtained by the updating mechanism for the hosts h, and in the same manner the initial risk scores $\bar{\mathrm{fr}}_{i+1}(f)$ of each flow f, from the corresponding values $\mathrm{fr}_{w_i}(f)$ and $\bar{\mathrm{fr}}_i(f)$:

$$\bar{\mathrm{hr}}_{i+1}(h) = c_{hu}\mathrm{hr}_{w_i}(h) + (1 - c_{hu})\bar{\mathrm{hr}}_i(h) \tag{3.6}$$

$$\bar{\mathrm{fr}}_{i+1}(f) = c_{fu}\mathrm{fr}_{w_i}(f) + (1 - c_{fu})\bar{\mathrm{fr}}_i(f) \tag{3.7}$$

In the above updating equations, c_{hu} and c_{fu} are constants of $0 \leq c_{hu} \leq 1$ and $0 \leq c_{fu} \leq 1$ which determine relative impacts of the values from the current time window versus previous updating values. In other words, if c_{hu} and c_{fu} are large, the risk scores can change fast; if c_{hu} and c_{fu} are small, the risk scores will change more slowly from one window to the next [5].

4 Experimental Evaluation

In this section we present results of performance evaluation of our system in order to validate its effectiveness and efficiency.

[5] In our experiment these constants were to 0.5 to equally reflect the importance of risk scores of current time window and the values from the previous updating process. Moreover, the initial values of risk for all objects are set to zero at the very beginning of the operation of our system.

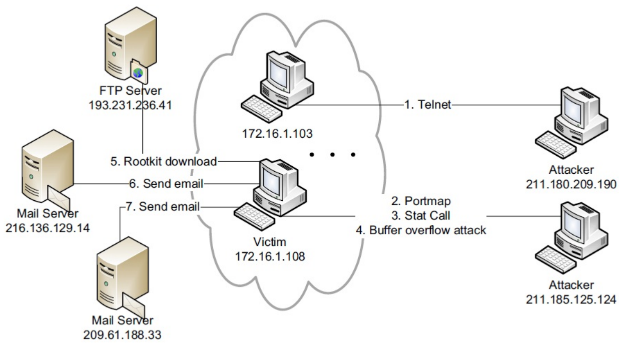


Figure 4.1: Attack scenario [19].

4.1 Experimental Environment

Our experiments are conducted on two public datasets which include two attack scenarios. To evaluate the effectiveness, we performed our risk computation on both of these datasets and show that the model assigned high risk scores to the victims and attackers which are involved in the attack scenarios. To evaluate the efficiency, we measure the elapsed time for our risk assessment process and two other recent models based on PageRank and HITS algorithms proposed in [20, 19], and we show that our model is superior to those two methods in terms of computational complexity. Moreover, we illustrate that the performance of our model is adequate for handling high throughput networks.

All the experiments have been conducted on an iMac PC with 2.00GHz Intel Core 2 Duo processor and 4GB RAM running Ubuntu 12.04 LTS. The program code has been written in Java with JDK 1.7. Table 4.1 summarizes the experimental parameters which are used for all experiments.

Table 4.1: Experimental parameters.

Parameter	Value
Maximum time for flow dependency	600 seconds
# of flows within a time window	2000
Length of sliding time window	20%
The source of risky ports	EMSISOFT
The maximum input cache size	10000
Accuracy threshold in iterative algorithm	0.001
Constants c_f, c_{hu} and c_{fu}	0.5
Constant c_{in}	0.25

Honeynet Dataset

In order to partially evaluate the effectiveness of our model, we performed the model on the public traces which were captured by the Honeynet project [15]. This scenario consists of a sequence of attacker activities including scanning, compromising, downloading and installing Rootkit, and sending spam emails (Figure 4.1).

MIT Lincoln Dataset

We also performed our risk computation model on MIT Lincoln dataset [6] which is used extensively for evaluating intrusion detection systems. Although there are several reservations regarding the accuracy of this dataset [13, 12], to our knowledge, no further datasets have been released.

The MIT Lincoln dataset includes a Distributed Denial of Service (DDoS) attack scenario and was originally proposed by DARPA project for evaluating Intrusion Detection Systems. The data file was collected over a span of approximately 3 hours on Tuesday, 7 March 2000, from 9:25 AM to 12:35 PM [6].

In the attack scenario of MIT Lincoln dataset, an attacker installs hacking tools on a number of internal machines. After that, the attacker performs a DDoS attack from all these compromised machines to the victim machine 131.84.1.31 by flooding packets. The statistical overview of the dataset is presented in Table 4.2.

Table 4.2: MIT Lincoln flow statistics.

The size of the pcap file	117M
Number of packets	649787
Capture duration	11652 seconds
Data bit rate	76680.78 bits/sec
Number of nodes (hosts)	34521
Number of flows	103006

4.2 Effectiveness

We evaluated the effectiveness of our solution by running the system on both Honeynet and Lincoln datasets. The Honeynet dataset has only 24 flow records and our program performed the risk analysis in a single window; experimental parameters are presented in Table 4.1. Some of the results of risk scores of hosts and flows on this dataset are presented in Table 4.3. These results show that our system assigned high risk values to hosts and flows which are related to the main victims and attackers (items in bold in Table 4.3). Therefore, the results validate effectiveness of our risk assessment method for small networks.

In order to evaluate the effectiveness of our risk assessment methodology for very large datasets, we applied our system to MIT Lincoln dataset. In order to perform risk assessment for a dataset of size 103006 flows (see Table 4.2) and with window size 2000 flows with 20% sliding factor (see Table 4.1), the program utilised 65 sliding time windows. At the end of each time window w_i, $1 \leq i \leq 65$, we have partial risk scores $\mathrm{hr}_{w_i}(h)$ and $\mathrm{fr}_{w_i}(f)$ for hosts and flows, respectively. In practice, such results, obtained in real time, would be used by the network administrator to monitor for possible malicious activity. We now evaluate these results for each time window as well as the final results after finishing the risk computation for all of the 65 windows.

Figure 4.2 illustrates the highest risk hosts detected by our system, based on the results in each time window. The bar chart in this figure shows for each

Table 4.3: Risk evaluation results for Honeynet Scan 18.

High risk flows		High risk hosts	
Risk	Flow	Risk	Host
0.08359145	**172.16.1.108:1026:193.231.236.41:21:tcp**	0.08534496	**172.16.1.108**
0.08205143	**172.16.1.108:1029:209.61.188.33:25:tcp**	0.0340215	**193.231.236.41**
0.07817169	193.231.236.41:1516:172.16.1.108:113:tcp	0.03213245	172.16.1.103
0.07817169	193.231.236.41:1519:172.16.1.108:113:tcp	0.02153438	209.61.188.33
0.07817169	193.231.236.41:1522:172.16.1.108:113:tcp	0.01920903	65.195.31.2
0.07673163	209.61.188.33:43497:172.16.1.108:113:tcp	0.01583514	172.16.1.107
0.07629695	172.16.1.108:1028:216.136.129.14:25:tcp	0.01526552	216.136.129.14
0.07287975	65.195.31.2:2473:172.16.1.103:53:tcp	0.01513408	172.16.1.106
0.07194606	211.180.229.190:3329:172.16.1.103:23:tcp	0.01490473	172.16.1.101
0.07175111	65.195.31.2:2477:172.16.1.107:53:tcp	0.0057409	211.180.229.190
0.0717024	65.195.31.2:2476:172.16.1.106:53:tcp	0.00495443	211.185.125.124

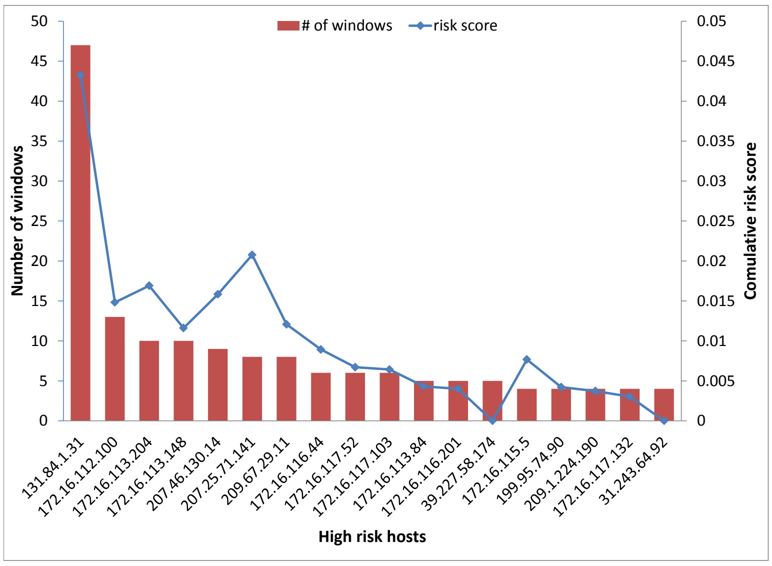


Figure 4.2: High risk hosts in Lincoln dataset.

host in how many windows (out of all 65 windows) was detected as a high risk. The line chart in this figure shows the cumulative risk computed for the highest risk hosts. One can see that in both metrics, our risk assessment model detected the main victim as the highest risk host (the host with IP address 131.84.1.31).

Moreover, we argue that the main objective of our risk assessment model is to rank the risk of network activities and therefore the proposed solution can not be replaced with an IDS. However, the results of our system can be applied as an input to security tools with deep inspection capabilities such as an IDS.

4.3 Efficiency

When comparing results with related research in [20] for Honeynet dataset, both approaches gave the highest risk values to attacker and victim nodes and flows. However, our approach has shown clear advantage in terms of its efficiency.

To explain better efficiency of our methodology, we recall that we use only one dependency graph for computing risk scores of both hosts and flows. On

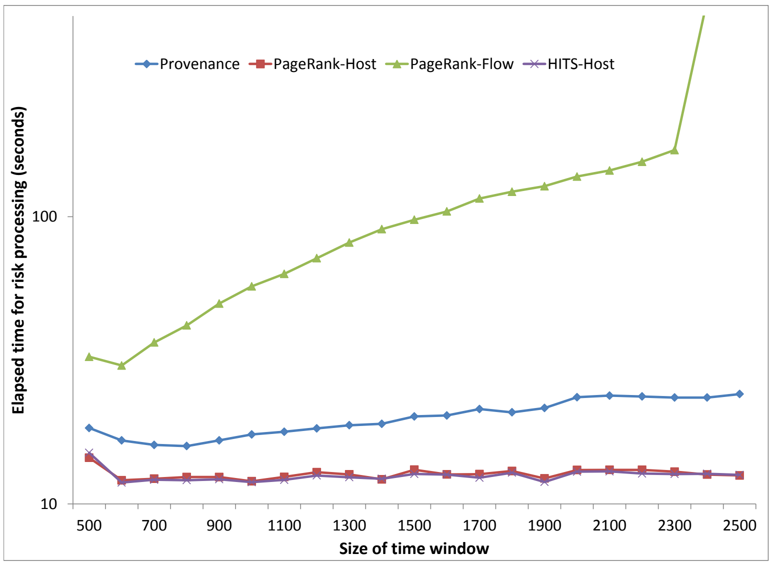


Figure 4.3: Elapsed time for risk computation techniques based on the size of time window.

the other hand, authors in [20] propose running link analysis algorithms on two graphs, one for hosts and another for flows. However, efficient computation of PageRank and HITS algorithms over even one very large dependency graph for a large scale network is a great challenge [1], let alone on two huge graphs.

As we discussed, the implemented sliding time window for input flows in our risk computation allows our approach to be deployed as an online monitoring tool for large scale networks. Moreover, for achieving more efficiency, a network administrator can alter the performance parameters of the program based on the provided hardware and the network throughput. The other advantage of our methodology stems from our use of sliding, overlapping windows; while this reduces the computation requirements for each window, our methodology does not suffer in effectiveness due to our use of the update mechanism which interconnects computations for successive windows, by passing the relevant risk information from one window to the next.

We quantify efficiency of our program by analysing its memory usage and processing time. The main parameter which influences the memory usage of the system is the maximum window size for the flow stream. Thus, we evaluate the processing time (the elapsed cpu time) of our provenance-aware algorithm along with risk assessment proposed in [19, 20] based on different size of time window. Figure 4.3 shows the results of this evaluation.

From Figure 4.3, we can see that in the algorithms proposed in [19] the elapsed time for computing risk scores of flows by applying PageRank algorithm increases dramatically as the size of time window increases (the green line). The reason is that the size of flow dependency graph is directly dependent on the size of the time window and the high processing time comes from applying PageRank on a very large graph. However, our provenance-aware mechanism computed both the scores of hosts and flows for such time window in a reasonable processing time (the blue line). Moreover, this figure shows that the processing time of our method increases much slower for large time windows and thus can handle high throughput networks.

As one can see in Figure 4.3, two linkage analysis algorithms based on PageRank and HITS for host risk assessment proposed in [20] require smaller processing time than our provenance based mechanism and same algorithm for flow risk assessment. The reason is that the size of a host dependency graph is significantly less than the size of a flow dependency graph [20]. In addition, these two methods only compute the risk scores for hosts, whereas our provenance-aware method obtains both score values for hosts and flows simultaneously. Computing both scores using the previous methods has a significantly higher time, approximately the sum of processing times presented by green and red lines on Figure 4.3.

5 Related Work

The related work to our research falls into two categories: dependency discovery among network traffic and data provenance management.

There are a number of papers investigating how to discover potential dependencies among network flows [4, 8, 9, 16]. Authors in [4] introduced the Orion system that discovers dependencies for enterprise applications by using packet headers and timing information. Iliofotou et al. in [8] proposed the use of Traffic Dispersion Graphs (TDGs) as a way to monitor, analyze, and visualize network traffic by modeling the social behavior of hosts. The key contribution in [9] is proposing a novel statistical rule mining solution, which was called eXpose, to extract significant communication patterns in a packet trace. Authors in [16] presented AssetRank, a generalization of PageRank algorithm, which automatically digests the dependency relations in an attack graph as well as the baseline information of the vulnerability attributes to compute the relative importance of attacker assets. While these works exploited dependency analysis on network activities, our method employs provenance relations among network flows as well as interdependency relation between hosts and flows in order to detect the high risk hosts and flows.

The most relevant works to our research is by Wang et al. [20, 19, 18], which aim at risk assessment for hosts and flows by employing link analysis algorithms such as PageRank and HITS. Authors in [19, 18] introduced flow dependency graph and applied PageRank and HITS algorithms to the graph in order to obtain risk scores of network flows. Due to the huge number of flows in a high throughput network and high computational cost of these algorithms on a very large graph, the efficiency of the idea in the network is a significant challenge. Therefore, the authors proposed host dependency graph in [20] and showed the new graph has less number of nodes and edges than the flow dependency graph. However, by applying the link analysis algorithms on the host dependency graph, they only obtain risk scores for hosts, without any decision about risk scores of flows. We propose the idea of flow provenance along with interdependency relationship between hosts and flows in order to evaluate efficiently the risk of both of them.

A large number of research work have been proposed for data provenance management [17, 3, 5, 11], but none of them deal with risk assessment of network flows based on flow provenance. Authors in [5], proposed a provenance-aware trust model for data management which takes into account various parameters that may affect the trustworthiness including data similarity, data conflict, path similarity and data deduction. Moreover, they considered the inter-dependency of trustworthiness between data items and the appropriate data provider. Also, they enhanced their trust model for sensor networks where the information keeps streaming into the system [11]. Our idea for proposing a provenance-aware model for risk assessment on network flows is inspired by the provenance-based trust model proposed by these two research work.

To the best of our knowledge, no existing work considers the provenance and interdependency between hosts and flows in order to assess risk on network activities. Different from the existing works, in this paper, we employ a novel, effective and efficient risk assessment solution for evaluating both risk scores of hosts and flows simultaneously.

6 Conclusions

This paper proposes a new method for risk assessment of network activities. The main idea behind our approach is to exploit the flow provenance and interdependency relationship between hosts and flows. The flow provenance is introduced based on flow dependency graph which takes into account the potential recursive causalities among network flows. The introduced interdependency relationship between the levels of risk of hosts and flows facilitates risk computation of both of them. We propose an iterative risk computation method based on these two properties. An update mechanism which is an integral part of our solution allows a deployment of our risk assessment methodology in real time. We have evaluated the effectiveness and efficiency of our method by performing the experiments on two publicly available datasets. Results show that our method is effective for assigning high risk scores to hosts and flows involved in attack scenarios as well as efficient in terms of processing time for performing risk assessment in a high throughput network.

As a future work, we plan to integrate our framework into OpenFlow and Software Defined Network (SDN) architectures [14] which can improve deployment of our system as an online monitoring system.

Bibliography

[1] Pavel Berkhin. A survey on PageRank computing. *Internet Mathematics*, 2(1):73–120, 2005.

[2] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. Link analysis ranking: algorithms, theory, and experiments. *ACM Trans. Internet Technol.*, 5(1):231–297, February 2005.

[3] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer Berlin / Heidelberg.

[4] Xu Chen, Ming Zhang, Z. Morley Mao, and Paramvir Bahl. Automating network application dependency discovery: experiences, limitations, and

new solutions. In *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, OSDI'08, pages 117–130, 2008.

[5] Chenyun Dai, Dan Lin, Elisa Bertino, and Murat Kantarcioglu. An approach to evaluate data trustworthiness based on data provenance. In *Proceedings of the 5th VLDB workshop on Secure Data Management*, SDM '08, pages 82–98, 2008.

[6] MIT Lincoln Laboratory: Cyber Systems & Technology: DARPA Intrusion Detection. Lincoln laboratory scenario (DDoS) 1.0, 2012.

[7] Emsisoft. Emsisoft portlist - all known TCP and UDP ports of malware, trojans, spyware, viruses. `http://www.emsisoft.com/en/kb/portlist/`, 2012. [Online; accessed 1-October-2012].

[8] Marios Iliofotou, Prashanth Pappu, Michalis Faloutsos, Michael Mitzenmacher, Sumeet Singh, and George Varghese. Network monitoring using traffic dispersion graphs (TDGs). In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 315–320, 2007.

[9] Srikanth Kandula, Ranveer Chandra, and Dina Katabi. What's going on?: learning communication rules in edge networks. *SIGCOMM Comput. Commun. Rev.*, 38(4):87–98, August 2008.

[10] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[11] Hyo-Sang Lim, Yang-Sae Moon, and Elisa Bertino. Provenance-based trustworthiness assessment in sensor networks. In *Proceedings of the Seventh International Workshop on Data Management for Sensor Networks*, DMSN '10, pages 2–7, 2010.

[12] Matthew V. Mahoney and Philip K. Chan. An analysis of the 1999 DARPA/Lincoln laboratory evaluation data for network anomaly detection. In *In Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection*, pages 220–237. Springer-Verlag, 2003.

[13] John McHugh. The 1998 Lincoln laboratory ids evaluation. In *Recent Advances in Intrusion Detection*, volume 1907 of *Lecture Notes in Computer Science*, pages 145–161. Springer Berlin Heidelberg, 2000.

[14] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.

[15] The Honeynet Project. Scan 18. `http://old.honeynet.org/scans/scan18/`, 2012. [Online; accessed 1-October-2012].

[16] Reginald E. Sawilla and Xinming Ou. Identifying critical attack assets in dependency attack graphs. In *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ESORICS '08, pages 18–34, 2008.

[17] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, September 2005.

[18] Shaonan Wang, Radu State, Mohamed Ourdane, and Thomas Engel. FlowRank: ranking NetFlow records. In *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, IWCMC '10, pages 484–488, 2010.

[19] Shaonan Wang, Radu State, Mohamed Ourdane, and Thomas Engel. Mining netflow records for critical network activities. In *Proceedings of the Mechanisms for autonomous management of networks and services, and 4th international conference on Autonomous infrastructure, management and security*, AIMS'10, pages 135–146, 2010.

[20] Shaonan Wang, Radu State, Mohamed Ourdane, and Thomas Engel. RiskRank: Security risk ranking for IP flow records. In *Network and Service Management (CNSM), 2010 International Conference on*, pages 56 –63, oct. 2010.

[21] Wikipedia. NetFlow — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/NetFlow`. [Online; accessed 1-October-2012].