Towards Effective Trust-Based Packet Filtering in Collaborative Network Environments

Meng, Weizhi; Li, Wenjuan ; Kwok, Lam-For

Published in:
IEEE Transactions on Network and Service Management

Link to article, DOI:
10.1109/TNSM.2017.2664893

Publication date:
2017

Document Version
Peer reviewed version

Towards Effective Trust-based Packet Filtering in Collaborative Network Environments

Weizhi Meng, Wenjuan Li, and Lam For Kwok

***Abstract*—Overhead network packets are a big challenge for intrusion detection systems (IDSs), which may increase system burden, degrade system performance and even cause the whole system collapse, when the number of incoming packets exceeds the maximum handling capability. To address this issue, packet filtration is considered as a promising solution, and our previous research efforts have proven that designing a trust-based packet filter was able to refine unwanted network packets and reduce the workload of a local IDS. With the development of Internet cooperation, collaborative intrusion detection environments (e.g., CIDNs) have been developed, which allow IDS nodes to collect information and learn experience from others. However, it would not be effective for the previously built trust-based packet filter to work in such a collaborative environment, since the process of trust computation can be easily compromised by insider attacks. In this work, we adopt the existing CIDN framework and aim to apply a collaborative trust-based approach to reduce unwanted packets. More specifically, we develop a collaborative trust-based packet filter, which can be deployed in collaborative networks and be robust against typical insider attacks (e.g., betrayal attacks). Experimental results in various simulated and practical environments demonstrate that our filter can perform effectively in reducing unwanted traffic and can defend against insider attacks through identifying malicious nodes in a quick manner, as compared to similar approaches.**

***Index Terms*—Intrusion Detection, Packet Filter, Trust Computation, Blacklist Generation, Collaborative Network.**

I. Introduction

NETWORK intrusions such as worms, Trojans and DDoS attacks are a big threat for computer networks and have already become more sophisticated to detect and defend [35]. For instance, *McAfee's threat prediction report* indicates that intrusions over the Internet would still be prevalent in future years [19]. The potential damage of these intrusions could be significant if they are not detected timely.

To address this problem, intrusion detection systems (IDSs) have been implemented at large with the purpose of defending against various attacks and they have become an indispensable component with respect to current defense mechanisms [29]. These detection systems usually identify an intrusion through comparing observable behavior against suspicious patterns. In particular, based on different detection methodologies, an IDS can be typically classified as *signature-based IDS* and *anomaly-based IDS*. A signature-based IDS (or *rule-based IDS*) (e.g., [25], [28]) detects a potential attack by comparing incoming events with its stored signatures, where a signature is a kind of description that defines an attack or an exploit by means of expert knowledge. On the other hand, an anomaly-based IDS (e.g., [13], [38]) tries to identify great deviations between current events and its pre-established normal profile. A normal profile often represents a normal action or a normal network connection through monitoring the normal behavior for a long period. In addition, based on the deployed locations and target events, an IDS can be classified as *host-based IDS (HIDS)* and *network-based IDS (NIDS)*. The former like [15] often resides on a local system and tracks changes made to important files and directories, while the latter like [37] usually places on the network with the purpose of analyzing network traffic for malicious patterns.

Motivations. Traditionally, an IDS often works in isolation so that it might be easily compromised by novel threats and complicated attacks (e.g., DDoS) [10]. Thus, collaborative intrusion detection networks (CIDNs) [41] have been developed, which allow a single IDS node within this network to collect useful information and learn experience from other IDS nodes, aiming to enhance the overall detection performance. Nowadays, IDS collaboration has become an effective way to facilitate the communications between detection nodes, and identify novel and complex attacks. However, IDS may also encounter various issues in such collaborative environment. In this work, we focus on two challenges: namely, *overhead network packets* and *effective trust computation*.

1) Overhead network packets. For a network-based IDS (especially a signature-based NIDS), overhead network packets are a very challenging issue. The term 'overhead' here means that incoming packets exceed the maximum handling capability of an IDS. In a large-scale network, massive amounts of incoming network packets can quickly exhaust computer resources, greatly decrease the performance of an IDS, and even cause the paralysis of the whole system [6]. Taking Snort [32] as an example, it usually spends around about 30 percent of its total computational power in conducting signature matching between the signatures and incoming packet payloads, while its computational consumption can be significantly increased in a large-scale network environment [9]. Typically, its computational burden is at least linear to the size of an input packet payload [27]. Previous research reports (e.g., [18], [30]) have indicated that an IDS cannot ensure the detection performance under the high-traffic environments. In the era of big data, this challenge will become more thorny and attractive.

W. Meng is with the Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark.
E-mail: weme@dtu.dk
W. Li is with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong SAR, China.
E-mail: wenjuan.li@my.cityu.edu.hk
L.F. Kwok is with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong SAR, China.
E-mail: cslfkwok@cityu.edu.hk

To tackle this issue, several approaches have been proposed such as the improvement of signature matching algorithms and pre-filtration of network packets. For the former, Schaelicke *et al.* [30] reported that it is still not sufficient to handle the overhead packet issue by only implementing a high quality algorithm in an IDS. Therefore, in our previous research, we advocated the filtration of network packets and had developed several packet filters that can complement those solutions in relation to signature matching improvement. More specifically, we first developed an adaptive blacklist-based packet filter to reduce packets via a statistic-based method [20], [23]. As the statistic-based method lacks of theoretical basis for generating blacklists, we then developed a trust-based blacklist packet filter using Bayesian inference [21], which was shown to be better than the statistic-based method in the aspects of blacklist false rates and traffic sensitivity.

2) Effective trust computation. In a CIDN, malicious nodes can greatly affect trust computation and decrease the effectiveness of packet filtration. As a result, the previously developed trust-based packet filter can perform well in a local system, but would not be effective to work in a collaborative network, since the process of trust computation can be easily compromised. In order to construct an effective trust-based packet filter, there is a need to evaluate a node's trustworthiness in a robust way, and defend against insider attacks (e.g., betrayal attacks) under a collaborative environment.

Contributions. In this work, motivated by the above challenges, we attempt to design a collaborative trust-based packet filter, which can provide an additional filtration mechanism in effectively reducing traffic under a collaborative network environment. To the best of our knowledge, our work is the first effort to develop a trust-based packet filter using Bayesian inference in a collaborative environment. Our contributions of this work can be summarized as below:

- We adopt the basic CIDN framework from the literature (e.g., [10], [11]), and modify it based on our scenarios. We then design a collaborative trust-based packet filter, which can conduct robust trust computation and effectively reduce target packets and workload for an IDS node[1]. It is worth noting that reducing the burden for an IDS node is very crucial for implementing a defense mechanism in real environments, like Ad Hoc networks.
- To improve the robustness and performance of our filter, we develop a collaborative process of trust computation for a node and an IP source, through collecting knowledge and learning experience from other trusted IDS nodes within the same network. *IP confidence* is used to represent the trustworthiness of an IP source.
- In the evaluation, we investigate the performance of our packet filter under an honest environment, a dishonest environment and a practical Ad Hoc network, respectively. Experimental results demonstrate that the packet filter can work well with low false rates of blacklist generation and promising packet reduction rates. Our proposed trust computation is found to be more robust against betrayal attacks (where a trusted node suddenly becomes malicious) than similar approaches.

Furthermore, we also discuss the scalability and CPU load of our collaborative packet filter through performing additional experiments. To clarify the scope of this paper, we limit our discussions on constructing an effective collaborative trust-based packet filter, whereas the improvement of collaborative intrusion detection (i.e., detecting zero-day attacks) is out of the scope. It is worth noting that our work aims to complement the existing studies in reducing the burden for IDSs.

The rest of this paper is organized as follows. In Section II, we introduce the previously built trust-based packet filter and review related studies on matching improvement, packet pre-filtration, collaborative IDS and trust management. Section III presents the modified CIDN framework and major components in detail. Section IV describes the construction of our filter and presents how to calculate the trustworthiness of a node and an IP source. Section V describes our evaluation including environmental settings, experimental methodology and collected results. We further discuss some relevant issues in Section VI and conclude the paper in Section VII.

II. Background and Related Work

In this section, we introduce the previously developed trust-based packet filter and review related work on signature matching improvement, packet pre-filtration, and trust management in distributed network environments.

A. Background of Single Trust-based Packet Filter

Previous studies (e.g., [4], [16]) have shown that constructing an appropriate filtration mechanism is a promising way to handle overhead packets. Motivated by this line of research, we developed a trust-based packet filter using Bayesian inference to help reduce a large number of network packets for a single NIDS [21], where the trust-based approach was proven to be better than a statistic-based method [20], [23]. The high-level architecture and deployment can be depicted in Fig. 1.

The trust-based packet filter has two major components: a *blacklist packet filter* and a *trust calculation engine*. The *blacklist packet filter* is mainly responsible for refining network packets based on their trust values (or called *IP confidence*). It consists of two special parts: a *blacklist* and a *look-up table*. The former contains all blacklisted IP addresses, while the latter contains NIDS signatures indexed by the blacklisted IP addresses. The *trust calculation engine* is used to collect data from both the blacklist packet filter and the deployed NIDS. With the data, this engine is responsible for evaluating IP trustworthiness and updating the blacklist periodically. The filtration steps can be described as below:

- If an incoming packet's IP address falls in the *blacklist*, then its payload has to be compared with the signatures stored in the *look-up table*. Note that the signatures here are the same as those in the deployed NIDS.
  - If a match is identified, then the *blacklist packet filter* will produce an alarm and send a copy of this alarm to the *trust calculation engine*.

[1]As CIDN is a network that consists of many IDS nodes, a node in this work mainly refers to an IDS node. Thus, this work will use terms 'node' and 'IDS node' interchangeably throughout the whole paper.

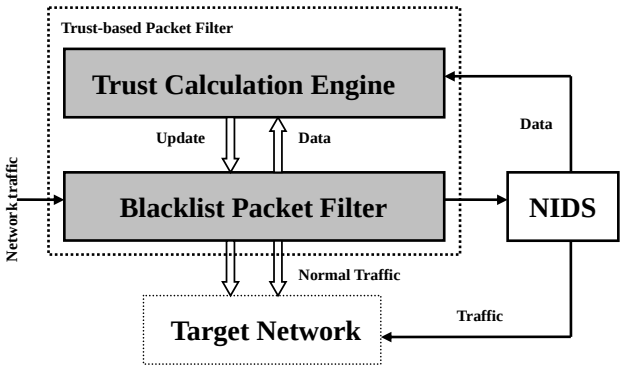

Fig. 1. The high-level architecture and deployment of a single trust-based packet filter.

  - If no match is found, then the packet will be treated as normal and be sent to the destination address.
- If an incoming packet's IP address is not found in the *blacklist*, then it will be forwarded to the deployed NIDS for examination.

In the component of *trust calculation engine*, *IP confidence* is designed to evaluate the trustworthiness of an IP source. To derive its calculation, we assume that N packets are sent from an IP source, of which k packets are proven to be *normal*, and that the distribution of observing $n(N) = k$ is governed by a Binomial distribution as below.

$$P(n(N) = k|p) = \binom{N}{k} p^k (1-p)^{N-k} \tag{1}$$

where $n(N)$ indicates the number of normal packets and p indicates the probability of a packet to be normal. Binomial distribution is the discrete probability distribution that represents the number of successes in a sequence of n independent, in which each n has the same probability of p. Hence the final goal of Bayesian inference model is to estimate the probability: $P(V_{N+1} = 1|n(N) = k)$ (determining whether the $(N+1)^{th}$ packet is normal or not). Based on the Bayesian theorem, we can have the following probability distribution.

$$P(V_{N+1} = 1|n(N) = k) = \frac{P(V_{N+1} = 1, n(N) = k)}{P(n(N) = k)} \tag{2}$$

where $P(V_{N+1} = 1|n(N) = k)$ indicates the possibility that the $(N+1)^{th}$ packet is normal if there are k normal packets out of N packets (see more details in [21]). For the above equation, we apply marginal probability distribution and can arrive at two following equations:

$$P(n(N) = k) = \int_0^1 P(n(N) = k|p) f(p) \cdot dp \tag{3}$$

$$P(V_{N+1} = 1, n(N) = k) = \int_0^1 P(n(N) = k|p) f(p) p \cdot dp \tag{4}$$

There is no prior information about $p \in [0, 1]$, so that we assume that p is determined by a uniform prior distribution $f(p) = 1$. Therefore, based on Equation (2), (3) and (4), we can have the following equation:

$$\begin{aligned} P(V_{N+1} = 1|n(N) = k) &= \frac{\int_0^1 P(n(N) = k|p) f(p) p \cdot dp}{\int_0^1 P(n(N) = k|p) f(p) \cdot dp} \\ &= \frac{k+1}{N+2} \end{aligned} \tag{5}$$

The experimental results in [20], [21] demonstrated that calculating trust values using Bayesian inference was generally better than using the statistic-based approaches in terms of both false rates of generating blacklists and traffic sensitivity. It is worth emphasizing that the packet filter would not affect the levels of network security provided by the deployed NIDS, due to the following circumstances [20]: 1) all targeted packets have to be compared with the stored signatures in the *look-up table*, which are the same as those utilized by the deployed NIDS; and 2) all non-targeted packets will be forwarded to the deployed NIDS for examination.

B. Related Work

1) Overhead Packet Mitigation: In the literature, signature (or string) matching improvement and packet pre-filtration are two major ways to manage overhead packets for typical intrusion detection systems.

Signature Matching Improvement. In order to deal with overhead traffic, one direct way is to improve the signature matching process. For single pattern matching, Boyer and Moore algorithm [2] is the most widely used scheme that utilizes two heuristics to reduce the number of searches in the matching process, where Horspool [14] provided an improved algorithm by using only the bad character heuristic. For multi-pattern string matching, Aho and Corasick [1] designed an algorithm that could search all strings at the same time through constructing a deterministic finite automaton (DFA) without the need for an additional search structure.

In the field of intrusion detection, Fisk and Varghese [9] first proposed an IDS-specific string matching algorithm called *set-wise Boyer-Moore-Horspool*. In the evaluation, their algorithm was demonstrated to be faster than both Boyer-Moor and Aho-Corasick algorithms, when handling medium size pattern sets. Some other related studies in relation to signature matching improvement can be referred to [3], [5], [17], [24], [31].

Pre-filtration. As Schaelicke *et al.* [30] revealed that designing an efficient matching scheme alone was not enough for overcoming overhead packets, many efforts have been made to reduce the burden of IDSs through either implementing matching schemes on hardware or pre-filtering network packets.

Song *et al.* [33] developed an FPGA-based pre-filter on hardware to reduce the amount of traffic for Snort, which consisted of a Xilinx VirtexE FPGA and a low-end PC running Linux. The hardware ensures that benign traffic passes through the system without the need for software processing. This allowed the system to operate on a 10 Gbps network. Ioannis *et al.* [16] then designed a packet pre-filtering approach on hardware to reduce the processing requirements for an IDS. They claimed that deploying the header matching portion of an IDS together with a small prefix match (in the range of 4-8

characters) could eliminate most of the rules and determine a handful of applicable rules, which may be checked more efficiently by a matching module.

Later, Chang *et al.* [4] provided two techniques to improve the FNP-like TCAM searching engine (FTSE) in high-speed networks, which was a two-stage architecture in detecting whether an incoming string contains patterns. The first approach performs pattern matching with a w-byte suffix instead of a w-byte prefix; and the second approach leverages the matching results from all groups excluding the first group. El-Atawy *et al.* [8] then presented an early filtering and decision technique, called *Relaxed Policy Expression*, which can reduce the packet matching cost by a dynamically changed pre-filtering phase (i.e., deploying before the original matching module). This technique mainly utilizes Internet traffic characteristics coupled with a special carefully tuned representation of the policy to generate early defense policies.

In this work, our developed collaborative trust-based packet filter is different from the former approaches and has its own features as below:

- The packet filter would not improve the signature matching process of an IDS, whereas our filtration mechanism can work with the existing pattern matching schemes and complement each other.
- Most previous filtration mechanisms attempt to analyze all incoming packets; however, our packet filter targets on the filtration of particular packets from the blacklisted IP addresses.
- Our packet filter is developed for a collaborative network environment, whereas most existing studies (e.g., [21]) are mainly discussed in a single IDS scenario.

It is worth emphasizing that our packet filter can be implemented on either software or hardware, aiming to complement existing filtration mechanisms. This is an early effort in constructing an effective trust-based packet filter using Bayesian inference in a collaborative network.

2) Collaborative intrusion detection and Trust Management: As mentioned earlier, IDSs have already become an essential defense mechanism in protecting a network against various threats. However, an isolated IDS has no information about the whole environment, so that it is more likely to be compromised by novel and complicated intrusions. As a result, collaborative intrusion detection networks (CIDNs) [41] have been developed aiming to enhance the detection accuracy of an IDS through collecting and learning required information from other IDS nodes. Relevant surveys in relation to CIDNs and coordinated attacks can be referred to [39], [42].

Trust management is very critical for CIDNs, as malicious nodes can significantly degrade network security and threaten benign nodes. To identify an insider attack in a collaborative network, Duma *et al.* [7] proposed a P2P-based overlay for intrusion detection (*Overlay IDS*) that could mitigate insider threats by using a trust-aware engine for correlating alerts and an adaptive scheme for managing trust. The trust-aware correlation engine was capable of filtering warnings that sent by untrusted or low quality peers, while the adaptive trust management scheme could use past experiences of peers to predict their trustworthiness. However, their approach has a major issue; that is, the past experience of a peer has the same impact regardless of the age of its experience.

To overcome this weakness, Fung *et al.* [10] first proposed a Host-based IDS collaboration framework that enables each IDS node to evaluate the trustworthiness of others based on its own experience via a *forgetting factor*. This factor can give more emphasis on the recent experience of the peer. Then, Fung *et al.* [11] improved their trust model by means of a Dirichlet-based approach, which could measure the level of trustworthiness among IDS nodes according to their mutual experience. This model has strong scalability properties and is robust against common insider threats. The experimental results demonstrated that the Dirichlet-based approach could enhance both robustness and efficiency. As feedback aggregation is a key component in a trust model, they further applied a Bayesian approach for feedback aggregation to minimize the combined costs of missed detection and false alarms [12]. Their experiments indicated that the Bayesian approach could make an improvement on the true positive detection rate and a reduction in the average cost.

In addition, Quercia *et al.* [26] further proposed a distributed trust framework with a risk-aware decision module, which satisfied a broader range of properties, i.e., evolving an expressive & tractable trust calculation based on Bayesian formalization, protecting user anonymity and enhancing detection of insider attacks. Other theories like information theory [34] and game theory [36] have also been used to evaluate the trustworthiness of communication entities in different domains.

With the advent of collaborative environments like CIDNs, the previously developed trust-based packet filter would not be effective, since its trust computation can be compromised by malicious nodes (e.g., betrayal attacks). In this work, our motivation is thus to design a collaborative trust-based packet filter, which can work effectively in a collaborative network and provide robust trust computation through calculating the IP trustworthiness in a collaborative way. Moreover, we adopt the theoretical CIDN model from the existing literature (e.g., [10], [11], [12]) and modify it according to our requirements. In the evaluation, we mainly compare our approach to similar trust models in defending against betrayal attacks.

III. CIDN Framework

To enhance the detection performance, a collaborative intrusion detection network enables an isolated or single IDS node to connect, communicate and cooperate with other IDS nodes. Traditionally centralized collaboration of IDSs often depends on a central server to gather information, but this server may become a single point of failure and the target of attacks. In this work, we thus adopt the well-established collaborative framework without a centralized server from [10], [11], [12]. Then, we modify it to fit our requirements through adding new developed components like *collaborative trust-based packet filter* and *collaboration component*.

A. Network join

As shown in Fig. 2, each IDS node can connect to other nodes and select its collaborators based on its own experience.

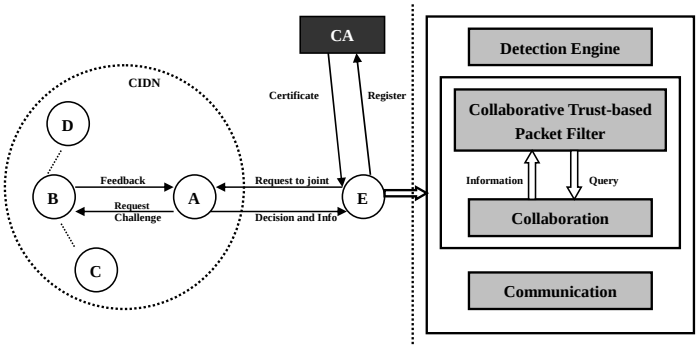

Fig. 2. The framework of our improved decentralized collaborative intrusion detection network.

Each node maintains a list of their collaborated nodes (called *partner list*). In the CIDN framework, if a new node E wants to joint the network, it needs to register to a trusted *Certificate Authority (CA)* and gets its unique proof of identity (including a public key and a private key). Different IDS nodes should have their unique certificates within the network.

After obtaining a valid certificate, node E can send a *joining request* to node A, so that node A is able to check and decide whether node E is acceptable based on the received certificate. If node A approves the request, it will send back a pair of *(decision, info)* to node E. *Decision* is a boolean value while *info* mainly contains an initial list of partner IDS nodes.

B. Request and challenge

In such framework, each IDS node can send out either *requests* for consulting IP confidence from others, or *challenges* to require other IDS nodes to give a response (or answer).

Specifically, *requests* are sent for collecting IP confidence from other IDS nodes. Such collected data can be used to calculate the value of *overall IP confidence*, which evaluates the trustworthiness of an IP source. By contrast, *challenges* are sent for evaluating the trustworthiness of IDS nodes in the *partner list*. For example, node A sends out a challenge to node B asking for the priority of an alarm, then node B has to send back a response. Based on the design methodology in [10], [11], [12], node A maintains its own alarm database and knows the expected answer to the response. Therefore, the trustworthiness of another IDS node can be measured between the received feedback and the expected answer.

To reduce traffic congestion, active responses are encouraged where a node will only reply to a number of requests in a certain period of time. Only highly trusted nodes have higher priority of sending and receiving feedback & requests. In addition, a reply of *unsure* is allowed in the communication for some inexperienced nodes.

C. Components

As shown in Fig. 2 (right half), an IDS node mainly consists of four components: namely, a *detection engine*, a *collaborative trust-based packet filter*, a *collaboration component* and a *communication component*.

- *Detection engine.* Within a CIDN, each node implements a signature-based IDS plugin like *Snort* [32] to identify attacks. This detection engine can be deployed behind the packet filter for traffic inspection.
- *Collaborative trust-based packet filter.* This packet filter attempts to narrow down the scope of traffic and reduce the workload for detection engine. In particular, it can calculate the value of *overall IP confidence* by considering the information collected from other trusted IDS nodes. When requesting an IP confidence from other nodes, it can forward a *query* to the *collaboration component*.
- *Collaboration component.* This component is responsible for assisting an IDS node in evaluating the trustworthiness of other nodes through sending out *requests* or *challenges* periodically, and collecting the corresponding *feedback*. When receiving a *query* from the packet filter, it can help send out a *request* to other nodes.
- *Communication component.* The goal of this component is to connect with other IDS nodes, provide network organization and management, and maintain communications between different nodes.

IV. Design of Collaborative Trust-based Packet Filter

In this section, we present the packet filter construction and describe how to compute the trustworthiness of a node (e.g., trusted or not) and an IP source (e.g., blacklisted or not).

A. Construction and deployment

Fig. 3 illustrates how to construct and deploy the collaborative trust-based packet filter. There are three major components: a *blacklist packet filter*, a *trust calculation engine* and *a collaboration component*.

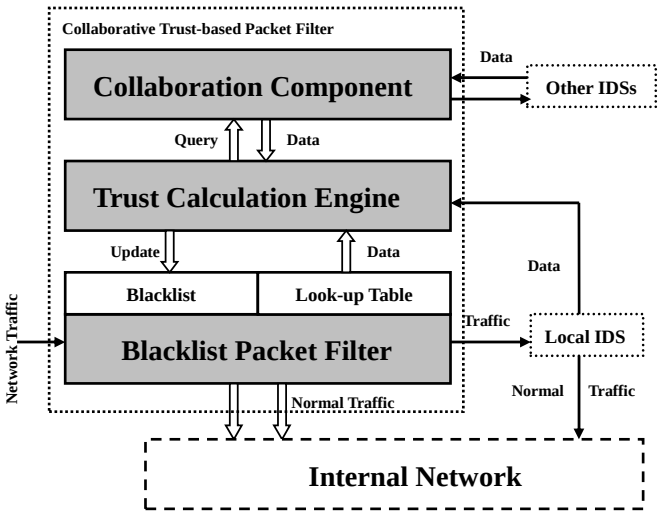

Fig. 3. The construction and the deployment of our collaborative trust-based packet filter, which consists of a blacklist packet filter, a trust calculation engine and a collaboration component.

Blacklist Packet Filter. This component is mainly responsible for refining network packets based on the IP confidence. As described in Section II, it consists of a *blacklist* and a *look-up table*. The blacklist contains all blacklisted IP addresses while

the look-up table contains all IDS signatures indexed by the blacklisted IP addresses.

Trust Calculation Engine. This component is responsible for collecting data from the blacklist packet filter, the deployed IDS and the collaboration component, computing the overall IP confidence and updating the blacklist periodically. The interactions among this engine, the local IDS and the blacklist packet filter are similar to the previous work [21]. Differently, in a collaborative network, this engine can send out a *query* to other nodes for collecting the IP confidence. The *collaboration component* will help collect the corresponding feedback and forward data to this engine.

Collaboration Component. This component is responsible for collecting the data (e.g., IP confidence) from other nodes and forwarding the required information to the *trust calculation engine*. When this component receives a *query* from the *trust calculation engine*, it will help send out a *request* to the target node and collect the relevant feedback.

B. Trust Computation

In this work, we develop two types of trust values: *node trust* and *overall IP confidence*. *Node trust* is used to evaluate the trustworthiness of a node, while *overall IP confidence* is used to evaluate the trustworthiness of an IP address, which can help generate a blacklist accordingly.

Node trust. In a collaborative environment, this kind of trust value aims to evaluate the trustworthiness of a node. According to the basic CIDN framework [10], [11], [12], the trustworthiness of a node can be calculated based on its responses to challenges. The *challenges* are sent out periodically by means of a random process. After receiving an *answer* to a *challenge*, a satisfaction level can be computed through identifying the gab between the received feedback and the expected answer. It is worth noting that the feedback from a node is ordered from the most recent to the oldest according to t_k. As a result, the trust value of node i according to node j can be computed as below:

$$T^{i,j}_{node'} = w_s \frac{\sum_{k=0}^{n} F_k^j \lambda^{t_k}}{\sum_{k=0}^{n} \lambda^{t_k}} \tag{6}$$

where $F_k^j \in [0,1]$ is the score in relation to the received feedback k, n is the total number of feedback, $\lambda \in [0,1]$ is a *forgetting factor* that assigns less weight to older feedback response, w_s is a *significant weight* depending on the total number of received feedback, m is the number limit of received feedback. If the number of feedback is smaller than m, then $w_s = \frac{\sum_{k=0}^{n} \lambda^{t_k}}{m}$; otherwise $w_s = 1$. Note that m aims to encourage those nodes with high request frequency to send back answers, and its value can be adjusted based on the high request frequency of challenges.

Additionally, based on the CIDN design from [10], [11], *'unsure'* answers are acceptable from an IDS node, when this node has no information about the challenges. This mechanism is especially beneficial for a newly joined IDS node that has no experience for the environment. However, a malicious node may use this mechanism to maintain its trustworthiness, there is a need to punish a node if it sends too many *'unsure'* answers. Thus, the trustworthiness of node i according to node j can be further calculated as below [10]:

$$T^{i,j}_{node} = (T^{i,j}_{node'} - T_{inital})(1-x)^y + T_{inital} \tag{7}$$

where $x \in [0,1]$ is the percentage of *'unsure'* answers from time t_0 to t_k, y is a positive parameter to control the severity of punishment on such answer, which can be computed based on the received feedback. T_{inital} is the default trust value of a new node. According to the above equation, a node's trust value would decrease, if it sends a large number of *'unsure'* answers (namely a large x).

Overall IP confidence. As mentioned above, *IP confidence* is used to evaluate the trustworthiness of an IP source. In a collaborative network, trust computation can consider the data from other trusted IDS nodes in addition to the information from the node itself. To achieve this, we define a metric of *overall IP confidence (T_{oic})* to represent the overall trustworthiness of an IP source. In terms of this metric and a threshold, the packet filter can reduce unwanted packets accordingly. If a node j wants to evaluate a target IP, then $T^j_{oic}(IP)$ can be calculated as below:

$$T^j_{oic}(IP) = \frac{\sum_{T^{i,j}_{node} \geq r} T^{i,j}_{node} D_i^j T^i_{oic}(IP)}{\sum_{T^{i,j}_{node} \geq r} T^{i,j}_{node} D_i^j} \tag{8}$$

where r is a trust threshold, so that node j has to request the information (e.g., IP confidence) from those nodes whose trust values are higher than r. $T^{i,j}_{node} (\in [0,1])$ indicates the trust value of node i according to node j. $D_i^j (\in [0,1])$ is a measure of *geographical distance* between node i and node j. $T^i_{oic}(IP) (\in [0,1])$ describes the IP confidence of a target source, which is computed by node i.

It is worth noting that $T^{i,j}_{node}$ here can also be considered as a *weight value* added to $T^i_{oic}(IP)$. Intuitively, a highly trusted node should have a larger impact on the calculation of *overall IP confidence*, as compared to other nodes. Thus, $T^i_{oic}(IP)$ can be extended based on Equation (5) as below:

$$T^i_{oic}(IP) = \frac{1 + \sum_1^n \lambda^{t_k} k_{t_k}}{2 + \sum_1^n \lambda^{t_k} N_{t_k}} \tag{9}$$

where n indicates the number of trusted IDS nodes in the partner list of node i and t_k denotes the time elapsed. Based on Equation (8) and (9), we can calculate the value of *overall IP confidence* for given IP addresses. If a threshold is set to $T \in [a,b]$, then we can decide a blacklisted IP address to be maintained or deleted via the following rules:

- If $T^j_{oic}(IP) \geq T$, then the blacklisted IP address will be deleted from the blacklist.
- If $T^j_{oic}(IP) < T$, then the blacklisted IP address will be maintained in the blacklist.

C. Satisfaction Mapping

According to the CIDN design in [10], [11], [12], a *challenge* can contain an IDS alarm and the *answer* is the priority of that alarm. As each node maintains an alarm database, the testing node can know the expected answer in advance. Thus,

the satisfaction levels can be measured by identifying the gap between the expected answer and the received answer. The satisfaction mapping will be described next.

Selection of Challenges and Answers. Based on the CIDN design in [10], [11], [12], an IDS node can download an alarm database from a server when joining the network. Then, the node has to maintain and update the database by itself (i.e., adding or updating items in the database based on its own experience). To evaluate other nodes, a *challenge* could be selected from this database via a random process.

When receiving a *challenge*, the tested node should search its own database and provide an answer. Based on the adopted assumptions from [10], [11], an honest node will always give the true alarm priority. That is, it will give the corresponding answer if a matched item is detected in the database; otherwise, it will give an 'unsure' answer. In contrast, a malicious node will always give a wrong answer.

Example. Taking *Snort* as an example, it has three alarm priorities: low, medium and high. Let R denote a mapping function, which maps an alarm priority to an interval of [0,1], where 0 denotes the lowest priority and 1 denotes the highest priority. If a testing node i sends a *challenge* containing an alarm A, then the tested node j should give an answer $R_j(A)$. Then, node i can compute its satisfaction based on the received answer $R_j(A)$ and its expected answer $R_i(A)$. For a more specific case, assuming node i sends a *challenge* including a *Snort* alarm as below:

[1:1201:8] ATTACK-RESPONSES 403 Forbidden [**] [Classification: Attempted Information Leak] [Priority: 2] TCP $137.245.85.134{:}80 -> 172.16.114.169{:}1049$

After receiving the *challenge*, the tested node has to search its own alarm database. For an honest node, an answer of 'Priority: 2' will be sent back if a matched item is detected; otherwise, an 'unsure' answer will be sent. For a malicious node, a wrong answer like 'Priority: 1' or 'Priority: 3' will be provided. Intuitively, it is a worse situation if a higher priority alarm was given a lower priority. In this example, a punishment can be given to the wrong answer of either '1' (high) or '3' (low). More severe penalty should be given to the answer of '3' (low), as it degrades the alarm priority.

Satisfaction Mapping. To facilitate the comparison with similar approaches, our work adopts the same approach from the previous work (e.g., [11], [12]) for computing the satisfaction level. Suppose there are two factors: the expected answer ($e \in [0,1]$) and the received answer ($r \in [0,1]$). A function F ($\in [0,1]$) can be used to reflect the satisfaction level of the received answer, in terms of its deviation level to the expected answer as below:

$$F = 1 - \left(\frac{e-r}{max(c_1 e, 1-e)}\right)^{c_2} \quad e > r \tag{10}$$

$$F = 1 - \left(\frac{c_1(r-e)}{max(c_1 e, 1-e)}\right)^{c_2} \quad e \leq r \tag{11}$$

where c_1 controls the degree of penalty for wrong estimates and c_2 controls satisfaction sensitivity. A larger c_2 means a more sensitive level. In this work, we set $c_1 = 1.5$ and $c_2 = 1$ based on the simulation results in [11], [12].

V. Evaluation

This section introduces the CIDN environment with simulation settings, describes our experimental methodology and evaluates our collaborative trust-based packet filter under an honest environment, a dishonest environment and a practical collaborative network environment, respectively.

A. Simulation Settings

To construct a CIDN environment, we deployed a total of 30 nodes that were randomly distributed in a 5×5 grid region. We used *Snort* [32] as IDS plugin and deployed Wireshark [40] to collect packets' information during the experiments. Each IDS node can connect to other nodes and establish an initial *partner list* based on the distance. The initial trust values of all nodes (in the *partner list*) are set to $T_{inital} = 0.5$.

To evaluate the trustworthiness of all nodes in the *partner list*, each node sends out challenges randomly to other nodes with an average rate of ε. In particular, there are two levels of request frequency: ε_l and ε_h. The request frequency of a highly trusted or highly untrusted node is low, since it could be very confident about the decision for their feedback. By contrast, the request frequency of other nodes is high, as their trust values may be close to the threshold and thus need to be checked more often. The simulation parameters in relation to the CIDN environment are summarized in Table I.

TABLE I
SIMULATION PARAMETERS IN THE EVALUATION.

Parameter	Value	Description
λ	0.9	Forgetting factor
ε_l	5/day	Low request frequency
ε_h	12/day	High request frequency
r	0.75	Trust threshold
$T_{initial}$	0.5	Trust value for new comers
m	10	Lower limit of received feedback
y	0.3	Severity of punishment

This work employs three expertise levels for a node as: low (0.1), medium (0.5) and high (0.95). The expertise of an IDS can be simulated using a beta function as below:

$$\begin{aligned} f(p'|\alpha,\beta) &= \frac{1}{B(\alpha,\beta)} p'^{\alpha-1}(1-p')^{\beta-1} \\ B(\alpha,\beta) &= \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt \end{aligned} \tag{12}$$

where $p'(\in [0,1])$ indicates the probability of an intrusion examined by the deployed IDS, $f(p'|\alpha,\beta)$ indicates the probability that a node with expertise level l responses with p' to an intrusion examination of difficulty level $d(\in [0,1])$. A bigger value of l means a higher probability of correctly identifying an intrusion, while a bigger value of d means that an intrusion is more difficult to detect. Based on [10], [11], α and β can be defined as below:

$$\begin{aligned} \alpha &= 1 + \frac{l(1-d)}{d(1-l)} r \\ \beta &= 1 + \frac{l(1-d)}{d(1-l)} (1-r) \end{aligned} \tag{13}$$

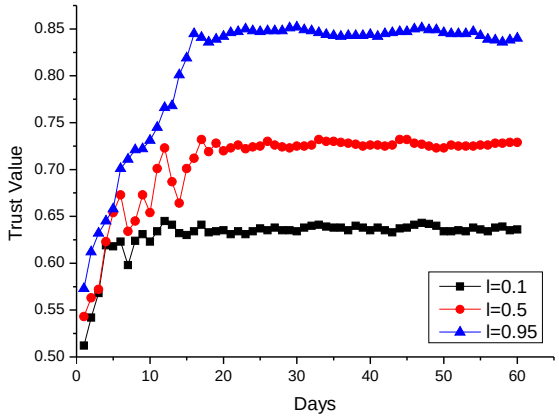

Fig. 4. Trust values vary with different expertise levels.

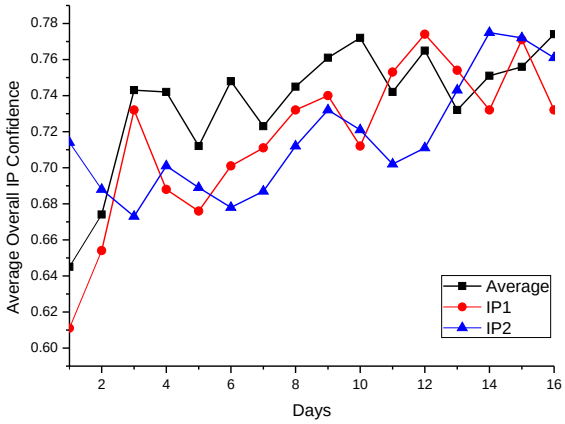

Fig. 5. The trend of average overall IP confidence for the external network.

where $r \in \{0, 1\}$ indicates the expected result of detection. For a fixed level of difficulty, the node with a higher expertise is able to achieve a higher probability of correctly identifying an intrusion. Specifically, a node with expertise level of 1 can accurately identify an intrusion with guarantee if the difficulty level is 0. This is reflected in the above Beta distribution with parameters $\alpha = 1$ and $\beta = 1$ (Uniform distribution). More details about Equation (12) and (13) can be referred to [10] and [11].

B. Experimental Methodology

In the evaluation, we conduct three experiments to investigate the performance of our designed collaborative trust-based packet filter in various scenarios. To facilitate the comparison with related studies, we accept the assumptions in relation to honest and malicious nodes from [10], [11].

- In the first experiment, we deploy the packet filter into an honest environment, in which an honest IDS node always generates feedback based on its truthful judgment. This experiment also attempts to find an appropriate threshold based on the false rates of blacklist generation.
- In the second experiment, we deploy the filter into a dishonest environment, where a dishonest IDS node always sends its feedback opposite to its truthful judgement. This experiment aims to evaluate the filter performance and the robustness of trust computation in a hazard scenario.
- In the third experiment, we investigate the practical performance of our filter in a real wireless Ad Hoc network (maintained by a company). The motivation is to explore its false rates of blacklist generation and packet filtration rate in a practical scenario.

C. Evaluation in an Honest Environment

This experiment attempts to investigate the performance of our designed packet filter in calculating *overall IP confidence*, and identify an appropriate threshold according to the false rates of generating a blacklist. To construct the environment, we randomly divided 30 nodes into three groups with different expertise levels: namely, low (0.1), medium (0.5) and high (0.95) respectively. In addition, we constructed an *external network* that was composed of 20 nodes. These external nodes with unique IPs can communicate with the CIDN nodes. In an honest environment, the trust values for different expertise nodes and the trend of *(average) overall IP confidence* for the external IP addresses are depicted in Fig. 4 and Fig. 5, respectively.

In Fig. 4, we simulate the first 60 days to investigate the trust values of different expertise nodes (l indicates the expertise level). It is visible that the nodes with higher expertise level can achieve higher trust values, in which the results conform to the findings in [10]. It is also noted that the trust values of all nodes could become stable after 17-18 days.

Fig. 5 describes the trend of *average overall IP confidence*, which is an average value for all 20 nodes from the external network. It is found that the value of *average overall IP confidence* gradually increased and ranged from 0.65 to 0.8. More specifically, we provide two examples of *overall IP confidence* for two external nodes, named *IP1* and *IP2*. It is visible that their IP confidence value ranged from 0.61 to 0.78 and from 0.67 to 0.783, respectively.

It is worth noting that generating a valid and robust blacklist is crucial for reducing unwanted traffic. Hence an appropriate threshold can be decided based on the performance of blacklist generation. In this work, we employ two metrics to evaluate the process of blacklist generation: *blacklist false positive rate (BFPR)* and *blacklist false negative rate (BFNR)*.

- BFPR: indicates the situation that a benign IP source is blacklisted.
- BFNR: indicates the situation that a malicious IP source is not blacklisted

Ideally, it is desirable to achieve a low value for both BFNR and BFPR; however, a balance should often be made in real deployment. Based on the results in [20], [21], it is found that the reduction of BFNR (i.e., examining more incoming packets) is more important than the reduction of BFPR (i.e., leaving packets to an IDS for examination), since the packet filter has the responsibility of reducing the burden for an IDS. As a result, an *appropriate* blacklist should have the following characteristics:

- *Accuracy.* The generation of a blacklist should achieve a low BFNR and BFPR (e.g., below 5%), after collecting

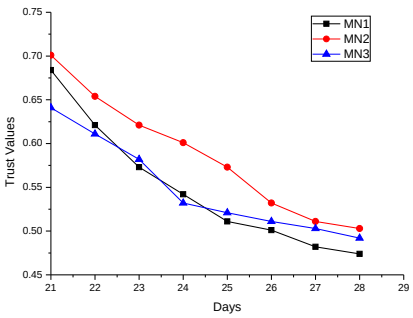

Fig. 6. Trust values for malicious nodes.

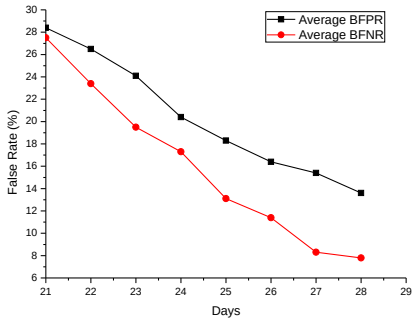

Fig. 7. Average false rates (BFPR and BFNR) in the dishonest environment.

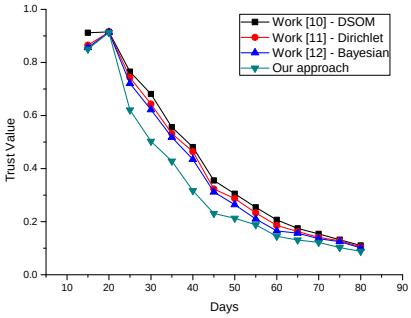

Fig. 8. Trust value of malicious node under betrayal attack.

traffic data for a period of time.

- *Effectiveness.* BFPR should not be smaller than BFNR, since the major goal of our filter is to help reduce the burden for an IDS node.

In order to identify an appropriate trust threshold, we mainly select and evaluate the influence of three values based on [10], [11]: namely 0.7, 0.75 and 0.8. The results of *blacklist false negative rate (BFNR)* and *blacklist false positive rate (BFPR)* are described in Table II (after 60 days).

TABLE II
RESULTS OF BFNR AND BFPR AFTER 60 DAYS.

Threshold	Average BFNR (%)	Average BFPR (%)
0.7	3.7	4.6
0.75	1.5	2.4
0.8	3.8	2.6

Table II shows that the average values of BFNR and BFPR vary with different thresholds. For the threshold value of 0.75, it is found that the lowest BFNR and BFPR can be achieved, and that BFNR is smaller than BFPR (which means that our filter can handle more incoming packets). In terms of both *accuracy* and *effectiveness*, we thus select the threshold value of 0.75 in the evaluation.

D. Evaluation in a Dishonest Environment

This experiment attempts to study the filter performance in a hazard situation, where some CIDN nodes become dishonest. In particular, we randomly selected a total of 10 expert nodes to be dishonest, since such nodes have the largest impact on the performance. The network settings were the same as the first experiment. The trust values of three malicious nodes (named *MN1*, *MN2* and *MN3*) are depicted in Fig. 6.

It is seen that the trust values of malicious nodes can be rapidly decreased, i.e., after Day 28, the trust value of *MN1*, *MN2* and *MN3* decreases from 0.68 to 0.47, from 0.7 to 0.5 and from 0.64 to 0.49, respectively. These results indicate that our method of trust computation can identify malicious nodes and decrease their trust values in a fast manner.

To validate blacklist generation, we randomly selected a total of 10 nodes from the external network to deliver malicious packets. The average values of *blacklist false negative rate (BFNR)* and *blacklist false positive rate (BFPR)* are depicted in Fig. 7. It shows that the false rates (including both BFNR and BFPR) were higher than 20% in the beginning, as the highly trusted expertise nodes send dishonest feedback to the packet filter. After collecting data and identifying malicious nodes, the false rates can decrease quickly (i.e., BFNR decreases from 27% to 7.8% and BFPR decreases from 28% to 13.6%). It is worth emphasizing that as long as the trust values of malicious nodes fall below the trust threshold, they cannot join and affect the calculation of *overall IP confidence* (since only highly trusted IDS nodes can participate).

1) Betrayal attack: This kind of attack is a potential threat for a collaborative network, in which a highly trusted node becomes a malicious one suddenly (i.e., infected by virus). To examine the performance of trust computation, we simulate a scenario: node *A* has seven nodes in its *partner list*, where only six are honest: two with high expertise, two with medium expertise and two with low expertise. The remaining one node also has high expertise, which first behaved honestly during the first 20 days, but conducted a betrayal attack afterwards. The trust value of such malicious node is shown in Fig. 8 (before and after the betrayal attack).

For the comparison of trust computation, we adopt three trust models from [10], [11], [12], since they are the most similar approaches. All these approaches employ a forgetting factor aiming to emphasize the more recent behavior of nodes, so that the trust value can drop smoothly. Under the betrayal attack, Fig. 8 shows that our approach can reduce the trust value of malicious node faster than other models. This because our approach is able to compute an overall trust value through considering both the local trust information, see Equation (5), and the information from other nodes, see Equation (9).

To summarize, a major difference can be identified between our approach and other trust management models in [10], [11], [12]. That is, our approach considers the state of packets (e.g., benign or not), see Equation (9), which is a continuous monitoring process and allows instant adjustment of trust values. In contrast, trust computation in [10], [11], [12] mainly depends on challenge-feedback mechanism, in which a *challenge* has to be delivered via a rate. Consequently, our approach is more sensitive to malicious behavior and can detect malicious nodes faster, as compared to other similar models.

In the evaluation, Fig. 6 and Fig. 8 validate that our method of trust computation works well in dishonest scenarios and is robust against malicious nodes (i.e., identifying malicious nodes in a fast manner).

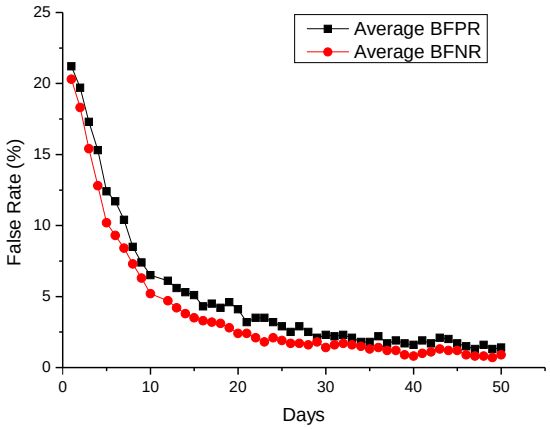

Fig. 9. The average false rates of generating a blacklist in the real network environment.

2) Node behaviour discussion: In practice, some malicious nodes can become normal again. For example, a node may turn malicious due to virus infection, whereas it can become normal again after cleaning up the virus. In this scenario, their trust values can increase as well. This situation is feasible and should be considered in a CIDN (see [10], [11], [12]). Under our adopted CIDN, it would be very time-consuming to increase a node's trust value than decrease it. In other words, even if a node become normal, its trust value can be decreased quickly if any malicious behavior is found again.

E. Evaluation in a Practical Network

In this experiment, we further deployed the packet filter in a real wireless Ad Hoc network (WANET) to investigate its practical performance. This network was established and managed by a company, where the incoming network traffic is about 1502 packets/s on average weekly, while the maximum speed of incoming traffic can reach 6823 packets/s. It was composed of 50 nodes and the basic settings can be adjusted according to Table I.

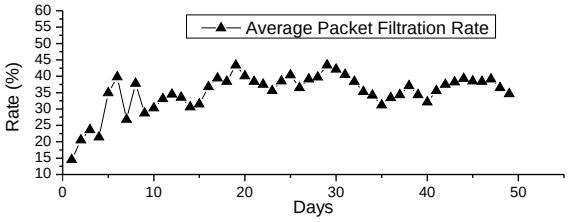

Fig. 10. The average packet filtration rate in the network environment.

In Fig. 9, we present the false rates (including both BFPR and BFNR) of generating a blacklist. In the beginning, both rates ranged from 20% to 22%, because the packet filter has to collect required traffic information in generating an effective blacklist that works on the deployed network. After a period of time, either BFPR or BFNR shows a downward trend, where they both decreased to 7% or below after 10 days. After another 10 days, both rates further decreased to below 5%, and gradually became stable (i.e., maintaining a value below 2%). As confirmed by the security administrators from the participating company, such false rates are acceptable and promising in the practical environment.

For the filter performance, Fig. 10 depicts *average packet filtration rate* under the practical environment. During the first four days, where the packet filter has to collect information in establishing a blacklist, the packet reduction rate ranged from 14% to 24%. Afterwards, the rate increased to 34.9% on the 5th day and then became relatively stable (i.e., above 33% in most cases). The rate fluctuation is mainly due to the blacklist update (i.e., deleting or adding IP addresses) and the communication delays. The filtering performance shows that our filter can perform well in effectively reducing target traffic for an IDS node under a practical network environment. This situation is also confirmed by the security administrators.

VI. Further Discussion

A. False Rate of Blacklist Generation

Due to the nature of network traffic and the communication delays, false rates are unavoidable for generating a dynamic blacklist. There are two types of false rates: namely, *blacklist false positive rate* (BFPR) and *blacklist false negative rate* (BFNR). It is worth emphasizing that *false rates* here can only affect the blacklist generation and filter performance, but cannot affect IDS performance and the whole network security. In other words, if we consider the filter as a black box, then the blacklist false rates are internal parameters, which can only make an impact on the filter itself. Furthermore, we explain the effect of both false rates as below:

- *Blacklist false positive rate*. A higher BFPR means that more benign IP addresses are blacklisted. Since the nature of a packet filter is to help reduce the unwanted traffic, a higher BFPR can indeed increase the filtration rate and reduce more workload for an IDS.
- *Blacklist false negative rate*. A higher BFNR means that more malicious IP addresses are not blacklisted. This can decrease the filtration rate, degrade the effectiveness of a packet filter and leave more packets to reach the deployed IDSs (i.e., causing a detection system ineffective).

Ideally, an appropriate filter should achieve a low value for both BFPR and BFNR (due to the requirement of *accuracy*), in which BFPR is expected to be higher than BFNR (due to the requirement of *effectiveness*). However, a balance should often be made in real scenarios. In the pratical environment (WANET), it is found that both false rates might be a bit high around 20% in the very beginning. This because there are no historical data provided in advance, so that the filter has to collect enough data in building an effective blacklist and a filtration profile for the deployed network. After a period of time, both false rates can significantly decrease (i.e., below 5% after 15 days as shown in Fig. 9).

B. Scalability

To validate the filter performance and scalability in different scenarios (e.g., with more nodes), we further deployed the filter in two distinct wireless Ad Hoc networks hosted by another company, which consisted of 60 and 70 nodes respectively.

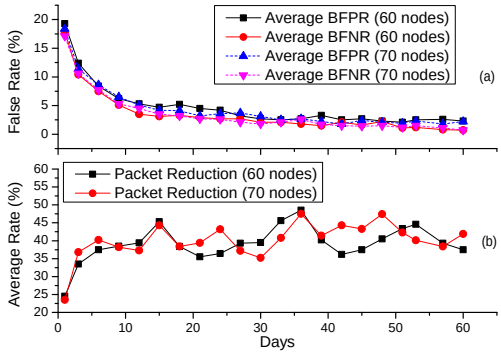

Fig. 11. (a) False rates of blacklist generation; and (b) packet reduction rates in different practical network environments.

The false rates (BFPR and BFNR) and packet reduction rates are depicted in Fig. 11.

Fig. 11 (a) indicates that both BFPR and BFNR ranged from 10% to 15% for the first three days, since the packet filter requires some time to collect traffic data and establish an effective blacklist. Afterwards, the false rates quickly decreased to 6% or below on the 10th day and then maintained a downward trend. After 20 days, the false rates became stable, where BFPR decreased to below 3% and BFNR decreased to below 1.5%. These results conform to the observations in our major experiments; that is, false rates of generating a blacklist can be reduced quickly after a period of time in collecting traffic data for the deployed environment.

Fig. 11 (b) shows that the average packet reduction rates ranged from 20% to 25% during the first two days. This because the packet filter had not established an effective blacklist in the very beginning. From the 3rd day, the reduction rate gradually increased to 35% or above, and maintained this trend in both network environments. As confirmed by the network managers from both participating organizations, the packet filtration rate here is considered to be promising.

The above experimental results demonstrate that our filter can achieve good scalability under various environments. More specifically, after collecting traffic data for a period of time, the filter is able to provide stable and low false rates of blacklist generation as well as a promising filtration rate.

C. CPU Load

Intuitively, implementing our filter, as an additional mechanism, may cause some CPU load during the packet reduction, due to various operations (e.g., packet forwarding and string matching). The workload is very crucial when deploying the filter in a resource-limited environment (e.g., wireless Ad Hoc network). Therefore, we aim to explore the CPU load caused by our filter and examine the relationship between the packet reduction and the CPU load reduction. For better illustration, we define several types of CPU load as below:

- *CPU_Load type1 (with the packet filter)*: measuring the CPU load caused by an IDS with the deployment of our packet filter.
- *CPU_Load type2 (without the packet filter)*: measuring the CPU load caused by an IDS without the deployment of our packet filter.
- *CPU_Load type3*: measuring the CPU load caused by the packet filter itself.

For simplicity, let CPU_{cpf} denote the CPU load caused by our packet filter (equal to *CPU_Load type3*), and CPU_{red} denote the overall reduced CPU load, which can be computed by identifying the difference of $|(type1+type3)-type2|$. The results of CPU_{cpf} and CPU_{red} in relation to the above three practical environments are depicted in Fig. 12.

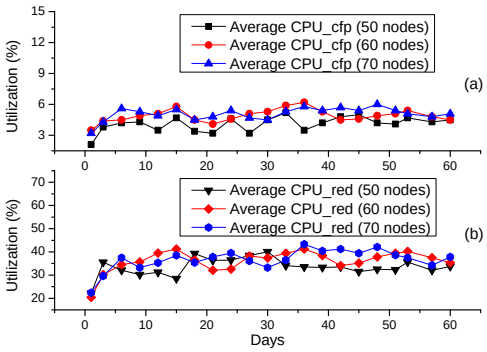

Fig. 12. (a) The average CPU load of our packet filter; and (b) the overall reduced CPU load with the deployment of our packet filter.

Fig. 12 (a) shows that our filter may add some overhead around from 2% to 6.2%, while Fig. 12 (b) indicates that our filter can greatly reduce the IDS workload ranged from 20% to 45%. This indicates that our packet filter can indeed reduce the overall CPU load for the whole system at large. Moreover, a relationship is identified between the packet reduction and the CPU load reduction, where a higher packet filtration rate can generally result in a higher CPU load reduction. As confirmed by the security officers from the corresponding organizations, our packet filter is proved to be effective in reducing the IDS workload in a practical environment.

D. Limitations and Challenges

This section discusses some limitations and open challenges in this area, such as IP spoofing attacks, the effect of filtering mechanism and database-size issue.

IP spoofing attacks. This kind of attack is a challenge for all list-based techniques, in which an attacker can manipulate a packet with a forged IP address. Since our filter employs the same signatures from the deployed IDS, all packets have to be compared to those signatures (a validation can be seen in [23]). As a result, IP spoofing attacks would not lower the detection rate of an IDS and the whole network security level. To defend against such attacks, an IP verification mechanism [22] can be used to validate the IP source and identify malicious ones.

The effect of filtering mechanism. It is worth emphasizing that the primary goal of our packet filter is to help reduce unwanted traffic and lighten the workload for an IDS in real scenarios. As a result, improving the detection accuracy of various network attacks like Byzantine attacks is out of the scope. These attacks can be identified through improving network protocols and deploying additional security mechanisms. As discussed above, no match found in the *look-up table* does not guarantee the examined packet is normal, but our filter

has no impact on the whole network security level. Moreover, our packet filter can complement and be compatible with the existing security solutions.

Database-size issue of look-up table and signatures. With the continuous running of our filter and the deployed IDSs, the number of signatures and the size of look-up table are likely to keep increasing, which may cause a database-size issue. To solve this problem, it is important to construct a database in a more effective way. This is an open problem and interesting topic, which can be considered in our future work.

VII. Conclusion

Overhead network packets are a big challenge for an IDS, where constructing a packet filter is a promising solution. With the advent of collaborative intrusion detection environments like CIDNs, it is found that the previously designed trust-based packet filter is not effective, as the process of trust computation could be easily compromised by insider threats, e.g., betrayal attacks, where trusted nodes suddenly become malicious.

In this work, our motivation is thus to design a collaborative trust-based packet filter, which can operate in a collaborative network (i.e., reducing the workload for an IDS node) and is robust against typical insider attacks (i.e., identifying malicious nodes). In the evaluation, we investigate the filter performance under an honest environment, a dishonest environment and a real network, respectively. Experimental results indicate that our filter can perform well in effectively reducing unwanted traffic and be robust against betrayal attacks through quickly detecting malicious nodes. Moreover, we conduct additional experiments to examine the false rates of blacklist generation, the filter scalability and the reduced CPU load. Overall, our filter can provide low false rates of blacklist generation (e.g., below 3%) and greatly reduce the IDS workload ranged from 20% to 45% in various scenarios.

Acknowledgment

The work was partially funded by the Innovation to Realization Funding Scheme of the City University of Hong Kong (under the project number 6351018).

References

[1] A.V. Aho and M.J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333-340, 1975.

[2] R.S. Boyer and J.S. Moore, "A fast string searching algorithm," *Communications of the ACM*, vol. 20, no. 10, pp. 762-772, 1977.

[3] A. Bremler-Barr and Y. Koral, "Accelerating multipattern matching on compressed HTTP traffic," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 970-983, 2012.

[4] Y.-K. Chang, M.-L. Tsai, and C.-C. Su, "Improved TCAM-based Pre-Filtering for Network Intrusion Detection Systems," *In: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (AINA)*, pp. 985-990, 2008.

[5] Y.-H. Choi, M.-Y. Jung, S.-W. Seo, "A fast pattern matching algorithm with multi-byte search unit for high-speed network security," *Computer Communications*, vol. 34, no. 14, pp. 1750-1763, 2011.

[6] H. Dreger, A. Feldmann, V. Paxson, and R. Sommer, "Operational Experiences with High-volume Network Intrusion Detection," *In: Proceedings of the 2004 ACM Conference on Computer and Communications Security (CCS)*, pp. 2-11. ACM, USA, 2004.

[7] C. Duma, M. Karresand, N. Shahmehri, and G. Caronni, "A Trust-Aware, P2P-Based Overlay for Intrusion Detection," *In: Proceedings of the 17th International Workshop on Database and Expert Systems Applications (DEXA)*, pp. 692-697, 2006.

[8] A. El-Atawy, E. Al-Shaer, T. Tran, and R. Boutaba, "Adaptive Early Packet Filtering for Defending Firewalls against DoS Attacks," *In: Proceedings of IEEE Infocom*, pp. 2437-2445, 2009.

[9] M. Fisk and G. Varghese, "An Analysis of Fast String Matching Applied to Contentbased Forwarding and Intrusion Detection," Technical Report CS2001-0670, University of California, San Diego, 2002.

[10] C.J. Fung, O. Baysal, J. Zhang, I. Aib, and R. Boutaba, "Trust Management for Host-Based Collaborative Intrusion Detection," *In: Proceedings of the 19th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM)*, pp. 109-122, 2008.

[11] C.J. Fung, O. Baysal, J. Zhang, I. Aib, and R. Boutaba, "Robust and scalable trust management for collaborative intrusion detection," *In: Proceedings of the 2009 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 33-40, 2009.

[12] C.J. Fung, Q. Zhu, R. Boutaba, and T. Basar, "Bayesian Decision Aggregation in Collaborative Intrusion Detection Networks," *In: Proceedings of the 2010 IEEE Network Operations and Management Symposium (NOMS)*, pp. 349-356, 2010.

[13] A.K. Ghosh, J. Wanken, and F. Charron, "Detecting Anomalous and Unknown Intrusions Against Programs," *In: Proceedings of the 14th Annual Computer Security Applications Conference (ACSAC)*, pp. 259-267, 1998.

[14] R. Horspool, "Practical fast searching in strings," *Software Practice and Experience*, vol. 10, no. 6, pp. 501-506, 1980.

[15] A. Hrivnak, "Host Based Intrusion Detection: An Overview of Tripwire and Intruder Alert," Technical Report, SANS Institute, January 2002. Available at: http://www.sans.org/reading_room/whitepapers/detection/host-based-intrusion-detection-overview-tripwire-intruder-alert_353

[16] S. Ioannis, D. Vasilis, P. Dionisios, and V. Stamatis, "Packet Pre-filtering for Network Intrusion Detection," *In: Proceedings of ACM/IEEE Symposium on Architecture for Networking and Communications Systems (ANCS)*, pp. 183-192, 2006.

[17] H. Kim, H.-S. Kim, and S. Kang, "A memory-efficient bit-split parallel string matching using pattern dividing for intrusion detection systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 11, pp. 1904-1911, 2011.

[18] W. Lee, J.B. Cabrera, A. Thomas, N. Balwalli, S. Saluja, and Y. Zhang, "Performance adaptation in real-time intrusion detection systems," *In: Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pp. 252-273, 2002.

[19] McAfee Labs, Threats Predictions in 2013. Available at: http://www.mcafee.com/us/resources/reports/rp-threat-predictions-2013.pdf

[20] Y. Meng and L.-F. Kwok, "Adaptive Context-aware Packet Filter Scheme using Statistic-based Blacklist Generation in Network Intrusion Detection," *In: Proceedings of the 7th International Conference on Information Assurance and Security (IAS)*, pp. 74-79, 2011.

[21] Y. Meng, L.-F. Kwok, and W. Li, "Towards Designing Packet Filter with A Trust-based Approach using Bayesian Inference in Network Intrusion Detection," *In: Proceedings of the 8th International Conference on Security and Privacy in Communication Networks (SECURECOMM)*, pp. 203-221, 2012.

[22] Y. Meng and L.-F. Kwok, "Enhancing List-based Packet Filter Using IP Verification Mechanism against IP Spoofing Attack in Network Intrusion Detection," *In: Proceedings of the 6th International Conference on Network and System Security (NSS)*, pp. 1-14 (2012)

[23] Y. Meng and L.-F. Kwok, "Adaptive Blacklist-based Packet Filter with A Statistic-based Approach in Network Intrusion Detection," *Journal of Network and Computer Applications*, vol. 39, pp. 83-92, 2014.

[24] D. Pao and X. Wang, "Multi-stride string searching for high-speed content inspection," *The Computer Journal*, vol. 55, no. 10, pp. 1216-1231, 2012.

[25] P.A. Porras and R.A. Kemmerer, "Penetration state transition analysis: A rule-based intrusion detection approach," *In: Proceedings of the 8th Annual Computer Security Applications Conference (ACSAC)*, pp. 220-229, 1992.

[26] D. Quercia, S. Hailes, and L. Capra, "B-Trust: Bayesian Trust Framework for Pervasive Computing," *In: Proceedings of the 4th International Conference on Trust Management (iTrust)*, pp. 298-312, 2006.

[27] R.L. Rivest, "On the Worst-case Behavior of String-Searching Algorithms," *SIAM Journal on Computing*, vol. 6, pp. 669-674, 1977.

[28] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," *In: Proceedings of the 13th Large Installation System Administration Conference (LISA)*, pp. 229-238, 1999.
[29] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," NIST Special Publication, pp. 800-894, 2007. http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf
[30] L. Schaelicke, T. Slabach, B. Moore, and C. Freeland, "Characterizing the Performance of Network Intrusion Detection Sensors," *In: Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pp. 155-172, 2003.
[31] R. Smith, C. Estan, and S. Jha, "XFA: faster signature matching with extended automata," *In: Proceedings of IEEE Symposium on Security and Privacy*, pp. 187-201, 2008.
[32] Snort: An an open source network intrusion prevention and detection system (IDS/IPS). Homepage: http://www.snort.org/
[33] H. Song, T. Sproull, M. Attig, and J. Lockwood, "Snort offloader: a reconfigurable hardware NIDS filter," *In: Proceedings of the 2005 International Conference on Field Programmable Logic and Applications*, pp. 493-498, 2005.
[34] Y.L. Sun, W. Yu, Z. Han, and K. Liu, "Information Theoretic Framework of Trust Modelling and Evaluation for Ad Hoc Networks," *IEEE Journal of Selected Areas in Communications*, vol. 24, no. 2, pp. 305-317, 2006.
[35] Symantec Corp., Internet Security Threat Report, 17 Main Report. http://www.symantec.com/business/threatreport/index.jsp
[36] T.A. Tuan, "A Game-Theoretic Analysis of Trust Management in P2P Systems," *In: Proceedings of the 1st International Conference on Communications and Electronics (ICCE)*, pp. 130-134, 2006.
[37] G. Vigna and R.A. Kemmerer, "NetSTAT: A Network-based Intrusion Detection Approach," *In: Proceedings of the 1998 Annual Computer Security Applications Conference (ACSAC)*, pp. 25-34, 1998.
[38] A. Valdes and D. Anderson, "Statistical Methods for Computer Usage Anomaly Detection Using NIDES," Technical Report, SRI International, January 1995.
[39] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and Survey of Collaborative Intrusion Detection," ACM Computing Surveys, vol. 47, no. 4, 2015.
[40] Wireshark: Network Protocol Analyzer. Homepage: http://www.wireshark.org/
[41] Y.-S. Wu, B. Foo, Y. Mei, and S. Bagchi, "Collaborative Intrusion Detection System (CIDS): A Framework for Accurate and Efficient IDS," *In: Proceedings of the 2003 Annual Computer Security Applications Conference (ACSAC)*, pp. 234-244, 2003.
[42] C.V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers & Securirty*, vol. 29, no. 1, pp. 124-140, 2010.

Weizhi Meng is currently an assistant professor in the Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Kongens Lyngby, Denmark. He received his B.Eng. degree in Computer Science from the Nanjing University of Posts and Telecommunications, China and obtained his Ph.D. degree in Computer Science from the City University of Hong Kong (CityU), Hong Kong in 2013. He was known as Yuxin Meng and prior to joining DTU, he worked as a research scientist in Infocomm Security (ICS) Department, Institute for Infocomm Research, Singapore, and as a senior research associate in CityU after graduation. He won the Outstanding Academic Performance Award during his doctoral study, and is a recipient of The HKIE Outstanding Paper Award for Young Engineers/Researchers in 2014 and the Best Student Paper Award from the 10th International Conference on Network and System Security (NSS) in 2016. His primary research interests are cyber security and intelligent technology in security including intrusion detection, mobile security and authentication, HCI security, cloud security, trust computation, web security, malware and vulnerability analysis. He also shows a strong interest in applied cryptography. He is a member of IEEE.

Wenjuan Li is currently a Ph.D. student in the Department of Computer Science, City University of Hong Kong (CityU). Prior to this, she worked as a Research Assistant in CityU from 2013 to 2014, and was previously a Lecturer in the Department of Computer Science, Zhaoqing Foreign Language College, China. She was a Winner of Cyber Quiz and Computer Security Competition, Final Round of Kaspersky Lab "Cyber Security for the Next Generation" Conference in 2014. Her research interests include network management and security, collaborative intrusion detection, spam detection, trust computing, web technology and E-commerce technology. She is a student member of IEEE.

Lam For Kwok received his Ph.D. degree in Information Security from Queensland University of Technology, Australia. He is currently an Associate Professor of the Department of Computer Science, City University of Hong Kong and the Executive Director of CUBIC (CityU Business and Industrial Club) at City University of Hong Kong. His research interests include information security and management, intrusion detection systems, and computers in education. He has served as program committee chair and program committee member of many international conferences. He is a Fellow of Hong Kong Institution of Engineers and British Computer Society.