SkipMon: A Locality-Aware Collaborative Intrusion Detection System

Emmanouil Vasilomanolakis*, Matthias Krügl*, Carlos Garcia Cordero*,
Max Mühlhäuser*, Mathias Fischer†
* CASED / Telecooperation Lab,
Technische Universität Darmstadt
{manolis, carlos.garcia}@cased.de, {matthias_johann.kruegl}@stud.tu-darmstadt.de
† Networking and Security Group,
International Computer Science Institute,
Berkeley, USA
mfischer@icsi.berkeley.edu

***Abstract*—Due to the increasing quantity and sophistication of cyber-attacks, Intrusion Detection Systems (IDSs) are nowadays considered mandatory security mechanisms for protecting critical networks. Research on cyber-security is moving from such isolated IDSs towards Collaborative IDSs (CIDSs) in order to protect large-scale networks. In CIDSs, a number of IDS sensors work together for creating a holistic picture of the monitored network. Our contribution in this paper is a novel distributed and scalable CIDS, called *SkipMon*. Our system supports, both, the idea of locality and privacy preserving communication by means of exchanging compact alert data. Furthermore, we propose a mechanism for interconnecting sensors that experience similar traffic patterns. The experimental results suggest that our CIDS, with our technique of connecting monitoring nodes that experience similar traffic, is scalable and offers a good accuracy rate compared to a centralized system with full knowledge of the participating sensors' data.**

I. Introduction

Sophisticated and highly tailored attacks, e.g., Distributed Denial of Service (DDoS) attacks and Advanced Persistent Threats (APTs), are constantly increasing [20]. Thus, Intrusion Detection Systems (IDSs) are nowadays considered mandatory for the protection and monitoring of critical infrastructures. However, isolated IDSs, i.e., systems that have no collaboration capabilities, cannot holistically monitor large-scale networks. For instance, targeted attacks, distributed scans and worm spreading cannot be tackled with such isolated IDS. For this reason, Collaborative IDSs (CIDSs), i.e., systems that collaborate by exchanging alert data to create a holistic view of the monitored network, have emerged [26]. CIDSs can provide scalability as well as the possibility to detect attacks that are widely scattered into different sub-networks.

For a CIDS to be efficient and usable in a practical manner, a number of requirements must be fulfilled [26], [29]. First, the system has to provide *scalability*, i.e., support for the monitoring of arbitrary network sizes. This should also be accompanied by a minimal message *overhead* as well as by high *accuracy*. Furthermore, we argue that such a system needs to be able to control the flow of alert network traffic in such a way so that only sub-networks that are allowed to communicate, can exchange messages. We define this requirement as *locality*, i.e., the ability to constrain alert dissemination, to certain sub-domains of a network, with respect to the ongoing security policy of a corporation.

978-1-4799-7575-4/14/$31.00 ©2015 IEEE

For instance, in such a corporate network different sub-networks might be logically separated due to a strict security policy. For example, the sub-network of the economics department may not be allowed to communicate with the development department, and so on. This *locality* property, to the best of our knowledge (cf. Section II) has not been addressed, so far, in the related work of CIDSs. This is important for the practical realization of such systems.

In this paper, we present *SkipMon*, a novel distributed CIDS approach that utilizes the SkipNet [6] Peer-to-Peer (P2P) overlay for the basic communication of its monitoring sensors. *SkipMon* offers two major contributions in the area of CIDSs. First, it supports locality, i.e., the ability to, on-demand, constrain the dissemination of alerts to certain sub-domains of the monitored network. Furthermore, we propose a novel mechanism for disseminating alert data and subsequently correlating the received information on the basis of bloom filters. In our system, sensor nodes exchange (alert) network traffic to discover others that experience similar traffic patterns. For this, we introduce a compact, privacy-preserving data dissemination mechanism via the utilization of bloom filters. Nodes that experience similar traffic subsequently create a community of nodes for exchanging more fine-grained alert data. In addition, *SkipMon* is open-source [5] and scales to large-scale networks.

We evaluate our system via the usage of real world network traffic to determine the messaging overhead, the accuracy of the suggested communities, as well as the effectiveness of the locality mechanisms. Our experiments indicate that *SkipMon* provides good accuracy rates into selecting the correct monitoring nodes that experience similar alert traffic. To evaluate this we compare *SkipMon* to a centralized system with full knowledge of the alert data of all the participating sensors.

The remainder of this paper is organized as follows. In Section II, we give an overview of the related work in the area of P2P overlays and CIDSs with a focus on distributed architectures. Section III, provides an extensive description of our system's architecture, and Section IV gives insights from our implementation. Subsequently, Section V presents results from the evaluation of *SkipMon*. Finally, Section VI concludes this paper and suggests ideas for future work.

II. Related Work

In this section we first provide some background knowledge regarding the P2P overlay that we will be utilizing and afterwards we discuss the related work in the area of distributed CIDSs.

A. P2P Overlays

Many proposals have been made over the years in the area of P2P networks and protocols [11], [13], [22]. With respect to CIDSs (cf. the next section) many of these distributed protocols have been utilized as a basis for creating a monitoring overlay. However, as we discuss in the following, none of the existing utilized P2P protocols can fulfill all the CIDS requirements [26]; especially the aforementioned practical need for locality.

Therefore, to meet fundamental requirements such as scalability and resilience (and also locality), we utilize the SkipNet P2P overlay [6] for the communication between monitoring sensors. SkipNet is an extension of SkipLists [17] for P2P networking. In this context two approaches have been proposed, i.e., Skip Graphs [1] and SkipNet [6]. We utilize the latter one as it provides features and details that are useful in a practical manner. Some of the notable properties that SkipNet offers are the routing tables that take into account the link quality between nodes and the network maintenance mechanisms that take place after major network disruptions.

In SkipNet all participating nodes are placed in a ring and identified with their reversed DNS name. In a practical realization this can be utilized to group nodes (in our case monitoring *sensors*) of the same organization or sub-network, as their hostnames end with the same domain names and their identifiers will start with the same prefix. Each sensor will not only have a link to their next neighbor but also to nodes 2^n hops away, similarly to the so-called *fingers* in Chord [22]. In this case, however, the nodes form sub-rings with the higher level links, as shown in Figure 1. For each routing level, nodes choose randomly the sub-ring that they will join. Their ring will lead to a second identifier, called numeric ID. Similar to SkipLists, each SkipNet node can hold data, which can be identified with a Uniform Resource Identifier (URI) providing the location and the name of the resource.

One of the benefits of SkipNet is the possibility to achieve *data* and *routing locality*. When storing data in SkipNet similarly to a Distributed Hash Table (DHT), data locality can be achieved. When specifying one node in the URI for a resource, the resource is stored at this node. Afterwards, the exact location of the resource is known. This is usually impossible when using load balancing. However, with the Constrained Load Balancing (CLB) in SkipNet, a domain can be specified, and the resource will be stored at a member of this domain. Thus, the location of the resource can be limited to groups of nodes; providing what we define as *routing locality*.

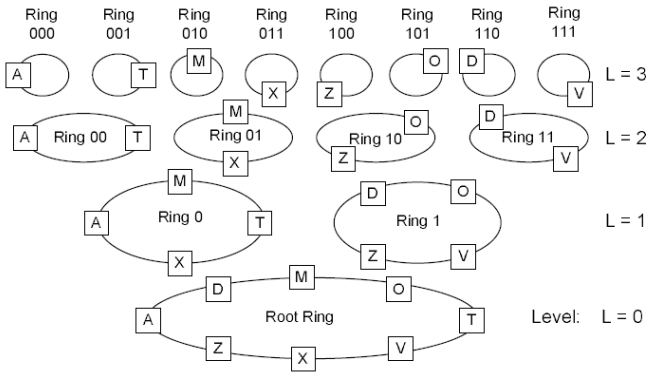

Fig. 1: SkipNet routing infrastructure example [6]

B. Collaborative Intrusion Detection

A lot of research has been conducted in the area of CIDSs over the last years [26], [29]. In more details, with respect to the utilized network architecture, a number of researchers have proposed centralized, e.g., [2], [19], [25] and hierarchical, e.g., [16], [21], [27], CIDSs. However, these systems experience fundamental architectural disadvantages. For instance, centralized CIDSs cannot scale to arbitrary network sizes, while hierarchical systems suffer from low accuracy as a result of the unavoidable correlation and aggregation of alert data (among the different levels of the hierarchy). Lastly, in both cases certain components might represent Single Point of Failures (SPoFs).

With regards to distributed CIDSs a number of proposals have been made [3], [7], [12], [28]. The majority of these approaches utilize P2P overlays, e.g., on the basis of DHTs, in which the monitoring sensors communicate the detected attacks in a collaborative manner. While these CIDSs are scalable they are usually limited in their detection capabilities (compared to a centralized CIDS) and/or they generate a considerable communication overhead. In addition, while DHTs are interesting from a research perspective, such flat overlays cannot be easily realized in a real-world environment. Thus, in contrast to the existing proposals, we focus on the P2P membership management component of the CIDS to make sure that only monitoring sensors that are experiencing similar traffic patterns are communicating. Moreover, the majority of existing work in such distributed CIDSs does not take privacy and locality into account.

A work that is relatively close to ours is from Locasto et al. [10]. The authors proposed the utilization of bloom filters as a means for privacy enhanced alert data dissemination. However, their approach suggested to randomly distribute bloom filters without taking into account any network traffic similarities.

They also did not give enough insights about their utilized P2P overlay, nor any information for its realization. Moreover, they did not take locality into account, but rather disseminated information in the whole P2P overlay.

To sum up, to the best of our knowledge no previous work has attempted to address locality and support this property as a requirement. In addition, *SkipMon* utilizes a deterministic data structure, i.e., bloom filters, to minimize the overall communicational overhead and ensure the privacy of the exchanged alerts. Lastly, this is one the first CIDSs that is open-source, while the implementation of the basic SkipNet P2P overlay itself is also one of the first ones.

III. SkipMon System Architecture

In this section we provide a detailed description of the *SkipMon* system by discussing its subcomponents. We utilize the architecture shown in Figure 2 to construct the five main building blocks that compose our system.

In more details, the *Local Monitoring* is responsible for the local detection as performed by the IDSs of each sensor. Sensors communicate by utilizing a P2P *membership management* protocol, i.e., the *SkipNet overlay*. Subsequently, sensors can exchange alert information by utilizing the *alert dissemination* mechanisms. In *SkipMon* we utilize a similarity-based *alert correlation* technique to identify sensors that experience similar traffic patterns. By utilizing such a mechanism it is possible to detect distributed port scans as well as malware propagation. Each sensor learns the traffic patterns of others and it is able to utilize a *community formation* algorithm. Afterwards, sensors can exchange more fine-grained alert information only with their community members. In the following subsections we detail each building block and how *SkipMon* fits in each block.

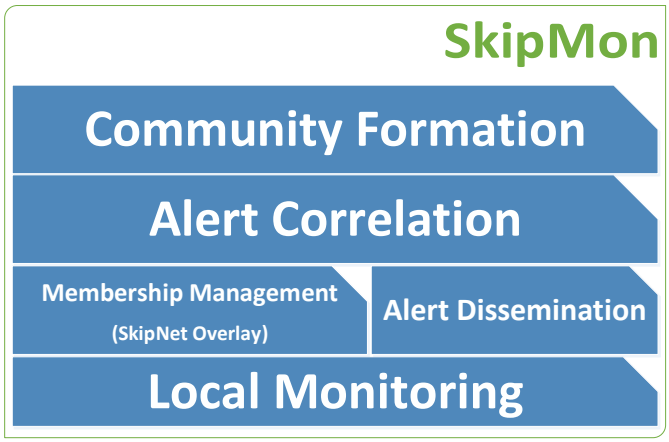

Fig. 2: High level architecture of *SkipMon*

A. Local Monitoring

A CIDS utilizes several IDSs to monitor an entire network. *SkipMon* is envisioned to make use of standard IDSs, e.g., Snort [18] or Bro [15] as long as they support standardized alert formats, e.g., the Intrusion Detection Message Exchange Format (IDMEF) [4]. In the current version of *SkipMon* we do not support this feature yet, and therefore we consider this out of the scope of this paper. Thus, in this architecture's block we assume that there are local IDSs capable of creating and disseminating local alert knowledge to other members of the CIDS.

B. SkipNet Overlay

For our system we make use of the SkipNet P2P overlay as introduced in Section II-A. As discussed, SkipNet can provide both *data* and *routing locality*. In this work, we focus on the routing locality to share information only with authorized sensors of a monitored network. Therefore, for *SkipMon*, data will be forwarded between the system instead of storing it at a given node or set of nodes. Due to the routing algorithms and ordering of nodes in SkipNet, data that shall be exchanged in one domain can and will only be routed through nodes of this domain. This leads to implicit data locality, as data transferred between two nodes in the same domain will never leave the boundaries of the domain in transit.

C. Alert Dissemination

To keep the communication overhead in the system low, alert information needs to be disseminated efficiently. Thus, we examine three alert dissemination techniques: *flooding*, *partial flooding,* and *gossiping*.

a) Flooding: As the name implies, each node sends messages to all neighbors in the SkipNet. Respectively, a node that receives a message will forward it to all of its neighbors. To minimize the overall communication overhead, redundant messages (i.e., messages that have been received from another node or path) are dropped. This is achieved by utilizing the message ID as we discuss in the next sub-section. Finally, it should be noted that to be able to flood the whole overlay, locality has to be deactivated.

b) Partial Flooding: In order to enable locality, we perform what we call partial flooding. Instead of exchanging messages with all possible neighbors, nodes exchange messages only with neighbors of the same sub-domain. This is possible because each message contains a locality value which can be used to query for neighbors in the same locality. Such a dissemination technique is particularly useful when security policies exist that prohibit the communication between different domains of a network.

c) Gossiping: Flooding creates significant network overhead that might exceed the available bandwidth and computational capabilities of CIDSs' sensors. Thus, we adapt the gossiping algorithm proposed by Kermarrec et al. [8], [9] and utilize it in *SkipMon*. The original algorithm uses a hierarchical communication approach where nodes are grouped into clusters. The communication links between nodes inside the same cluster are called *intracluster* links. The communication links that nodes within one cluster maintain to any other node outside of its cluster are called *intercluster* links. This work adapts these concepts to preserve locality within *SkipMon*.

Gossiping enables messages, with a probabilistic guarantee, to reach a subset of nodes in a network without flooding. The

probability of a message reaching all nodes within one cluster, that is, of every node in a cluster having a directed path to every other node within that same cluster, is given by [8]

$$p_n = \exp(-e^{-c_1}), \tag{1}$$

where n is the total number of nodes in the cluster, e is the Euler constant, and c_1 is a constant. By fixing p_n to a desired value and solving for c_1, it is later possible to determine the number of required *intracluster* links k that each node in the cluster needs, for disseminating information efficiently, using

$$k = \log(n) + c_1. \tag{2}$$

SkipMon is also concerned with the preservation of locality. This implies that not all nodes have the ability to contact or communicate with every other node outside of its locality. This is the same as restricting the number of *intercluster* links that exist between node clusters. If we consider each locality as a cluster, the number of *intercluster* links f required to guarantee a probability p_m of having all clusters (or localities) m connected with a path is defined as

$$f = \log(m) + c_2. \tag{3}$$

Once again, the constant c_2 can be calculated by fixing p_m and solving for c_2 in $p_m = \exp(-e^{-c_2})$.

D. Alert Correlation

In the following, we discuss the alert correlation in *SkipMon*. For this, we first discuss the construction of alert messages, and subsequently present our similarity-based correlation technique.

1) Alert Messages: The alert messages produced by local IDSs may contain a lot of redundant information for a CIDS.

To cope with this we make two important decisions in the context of representing alert messages. First, we argue that only a small fraction of the alert messages' data is required for other sensors to be able to discover similarities. Bearing this in mind, we utilize a number of important features for representing alerts, i.e., IP addresses of attackers, as well as source and destination port numbers. Note that this decision is taken in many CIDSs in related work [26].

To handle and exchange alerts in a compact and privacy-preserving way, we utilize bloom filters [23]. Bloom filters are a probabilistic data structure that represents elements in a set and provides an efficient mechanism to check whether a particular element is part of the set or not. Bloom filters can handle a very large amount of data in an efficient manner. In addition, they preserve privacy as no information can be leaked out. In fact, they only support checking whether a certain element is part of the set or not. Therefore, organizations can use CIDSs that support the distribution of messages via bloom filters without the fear of revealing sensitive information. Moreover, bloom filters do not produce false negatives and the false positives ratio can be adjusted with the following equation [14]:

$$P_{fp} = (1 - (1 - \frac{1}{m})^{kn})^k, \tag{4}$$

where m is the number of bits in the bloom filter, k the number of hash functions that are utilized, and n the number of elements in the bloom filter. The aforementioned properties, and especially Equation 4, are important as they depict the applicability of bloom filters in our concept.

We make use of the bloom filters in the following way. Each sensor produces alerts from which we extract features, e.g., the (adversaries) IP addresses, and subsequently add them into a bloom filter. Afterwards, each sensor will utilize the available alert dissemination techniques (cf. the previous subsection) and send their bloom filters to other nodes.

Fig. 3: Messages in *SkipMon*

Overall, messages in *SkipMon* contain four different fields, as shown in Figure 3. The first field, *bloom filter*, contains the actual bloom filter. In addition, the *sender node name* as well as the *message identifier* (i.e., a hash value) are added. Both of these fields are utilized for minimizing redundant messages and, thus, reducing the overall overhead when disseminating messages. Lastly, the *locality value* (L) is an integer that defines the depth of *SkipNet* sub-domains that the message can reach. For example, a *zero* value ($L = 0$) indicates that locality is disabled and the message can be disseminated to any sub-domain. A value of *one* ($L = 1$), however, would indicate that messages can be disseminated to a sub-domain if and only if the first field of the DNS name of two nodes is the same. An example, for three different L values, is also given in Figure 4.

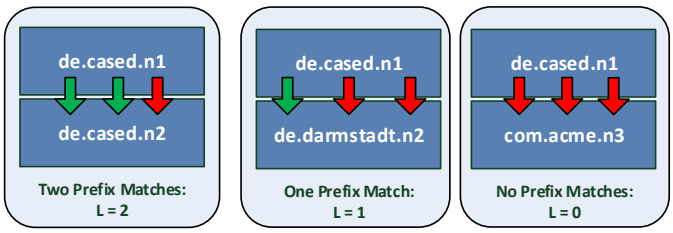

Fig. 4: Locality example in *SkipMon*

2) Similarity Correlation: Alert correlation takes place when a node receives a message to determine whether the received message and, specifically, its alerts are relevant for the recipient node or not. The goal of correlating alerts is to provide a mechanism for connecting nodes that experience similar traffic patterns. Regardless of the utilized alert dissemination technique, nodes receive messages from other nodes and compute their similarity value. For this, we make use of the inherent properties of the bloom filters and their ability to perform logical operations such as the bitwise AND ($\wedge$) and bitwise OR ($\vee$). To be able to do so, all bloom filters must have the same size and utilize the same hash functions.

We define the similarity $S_{a,b}$ of two nodes n_a and n_a as

$$S_{a,b} = \frac{bf_a \wedge bf_b}{bf_a \vee bf_b}. \quad (5)$$

Each node is represented by the set of bits found in their bloom filters. The similarity correlation of two nodes is calculated by dividing the bitwise AND over the bitwise OR of their set of bits.

After calculating the similarity value, nodes will make use of a threshold value t to determine whether S is similar enough or not. As we will discuss in the next section, the threshold creates a leverage in the number of proposed communities of sensors; when t is low, for example, a large number of sensors are found to be similar and therefore grouped together.

E. Community Formation

After the successful dissemination and correlation of the alert data, each sensor creates a matrix with its local knowledge of other sensors. Based on this knowledge and along with the utilized threshold, sensors can identify others and form a community with them to, afterwards, exchange more fine-grained alert data. An example of such a matrix is shown in Table I. In the case where the threshold $t = 0.8$, node 3 (n_3) would only create a community with node 4 (n_4). Nevertheless, details on the exchange of alert data during a community formation are out of the scope of this paper and will be our main consideration in our future work.

Node	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$S_{3,4}$	$S_{3,5}$
n_3	0.5	0.7	**1**	**0.9**	0.4

TABLE I: Example of similarity scores for node n_3

IV. Implementation

Our prototype [5] is written in C++, containing more than 6500 lines of code, and it is distributed under the GNU Lesser General Public License (LGPL) $v.3$. Figure 5 gives an overview of the architecture of our implementation. In more details it provides a detailed view on how different modules of our implementation are connected. In the following, we briefly discuss each of them, i.e., the *Control Module* the *Node Management*, as well as the *SkipNet/SkipMon* sub-modules.

The *Control Module* is responsible for managing multiple *Node Management* instances, monitor their status, as well as (for the purpose of the evaluation) injecting alert data to the nodes. The *Node Management* is responsible for reporting the status of *SkipMon* nodes (to the Control Module), for connecting sub-modules of the system and for providing an interface for exchanging routing information. The first sub-modules that are started from the Node Management are the *SkipNet* nodes (implemented following [6]). *SkipNet* nodes form an overlay that is used as a backbone for all the further operations that are done by the *SkipMon* sub-modules. The latter store alert data into bloom filters, share them in the network, and correlate information received from other nodes as discussed in the previous section.

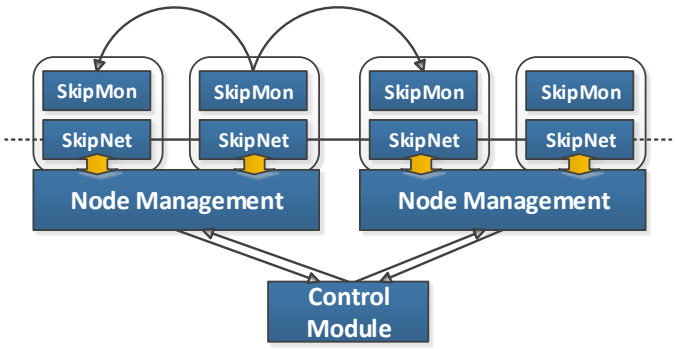

Fig. 5: *SkipMon* implementation overview

V. Evaluation

In this section, we provide insights and a discussion of the results gathered from our evaluation. We are interested on measuring the performance of our approach in terms of its accuracy (in the context of finding sensors that experience similar traffic patterns) compared to a centralized system with full knowledge of the alert data of all participating sensors.

We first describe our dataset as well as the evaluation setup. Afterwards, we discuss the results by studying the accuracy of our alert correlation technique and the locality properties of *SkipMon*.

A. Dataset Description

For evaluating our system, we utilized a dataset provided from DShield [24]. The DShield project collects alert data that is sent in by volunteers, e.g., IDSs and firewalls, from all over the world. In more details, we utilized data from a 24 hours period. The data of this day consists of $7,841,775$ alerts from $232,379$ unique attackers, reported by $138,192$ monitoring sensors.

Table II shows an excerpt of those data. The entry for an event is organized as follows:

- Log ID: A unique ID for each alert.
- Time stamp: The exact time stamp (date and time) in which an alert occurred.
- Source Port: The port that was used for the malicious activity.
- Target Port: The port that was targeted during the malicious activity.
- Protocol: The protocol number of the generated alert.
- IDS Hash: A unique ID (i.e., a 160 bit hash) that serves as a pseudonym for each IDS providing data (truncated in Table II).

The dataset was pre-processed, that is, the alerts were sorted with respect to the reporting sensor and stripped of any information other than the IP address. In addition, since IP addresses can occur multiple times per node, e.g. when a port-scanning multiple ports of the same sensor, duplicate IPs (targeting the same sensor) have been removed. Finally, only the data of the top contributing monitoring sensors was taken into account for the evaluation; sensors that provided less than 10 alerts were excluded from the evaluation.

Log ID	Time Stamp	Malicious IP Address	Source Port	Target Port	Pro-tocol	IDS Hash
461600985805	2015-05-06 18:38:15	116.211.000.090	50978	8080	6	8078...
450205301168	2014-01-01 02:51:47	094.247.233.129	40370	5900	6	C58A...
450205304225	2014-01-01 02:52:59	087.118.007.218	3487	445	6	FCBE...

TABLE II: DShield dataset example

B. Evaluation Setup

The purpose of our evaluation is to assess the accuracy of the mechanism of detecting similar sensors, to compare the different dissemination mechanisms, and lastly test how the locality property influences the accuracy of detection of similar sensors. For this, we discuss the results of 50 repetition runs, with 100 monitoring sensors that each of them contains a maximum of 1000 alerts in their bloom filters. The respective plots include the min/max values of the number of proposed communities for each threshold value.

For measuring the accuracy of detecting similar sensors we utilize a metric called *number of proposed communities*. This, as the name implies, refers to the number of (correctly) proposed communities of sensors and we examine it with respect to various similarity thresholds (by utilizing the Equation 5). To assess the accuracy of our proposal, we compare *SkipMon* to a centralized system that possesses global knowledge of all the alert data of the participating sensors. For a more detailed look on the differences of a centralized system with *SkipMon* we utilize the false positive and false negative metrics, that we define as follows. *False positives* refer to the communities of sensors that were proposed in our distributed system, but were omitted in the centralized system. Similarly, *false negatives* represent the communities of sensors that have been proposed by the centralized system, but were not detected by *SkipMon*. Moreover, for measuring the communicational overhead we count the total number of exchanged messages.

C. Results

In the following a detailed discussion of the results is given with regards to the accuracy of the alert correlation and the locality properties.

1) Accuracy of alert correlation: For disseminating alerts in *SkipMon*, we utilize the flooding and gossiping mechanisms described in Section III-C. More specifically in the case of gossiping we make use of Equations 2 and 3 by setting the probability $p_n = 0.9$.

Figure 6 presents the results of flooding and Figure 7 the results of gossiping respectively. As one can observe the accuracy for both techniques is close to the centralized system. Moreover, as expected, the numbers of proposed communities in all cases significantly decrease when the threshold is increasing. Lastly, Table III depicts an overview of the communicational overhead, by counting the total number of messages that are exchanged in each of the three cases. The centralized system requires a lower number of messages but this metric does not take into account the computational overhead for the central component, or the need for scalability.

CIDS	Mean number of messages	Min	Max
Centralized System	452	452	452
SkipMon (flooding)	1812	445	4793
SkipMon (gossiping)	1346	290	2030

TABLE III: Communication overhead comparison

In addition, as expected flooding generates significantly more messages than gossiping.

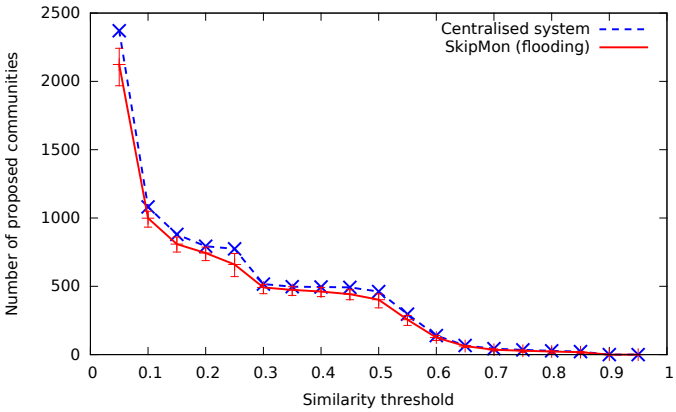

Fig. 6: Proposed communities by *SkipMon* (with flooding) compared to a centralized system

For a more detailed look on the dissemination mechanisms in *SkipMon* we examine the false positive and false negative metrics. Figure 8 presents the results of false positives and negatives when flooding and Figure 9 when using gossiping. On the one hand, the amount of false negatives in the case of flooding can be attributed to information loss in the system,

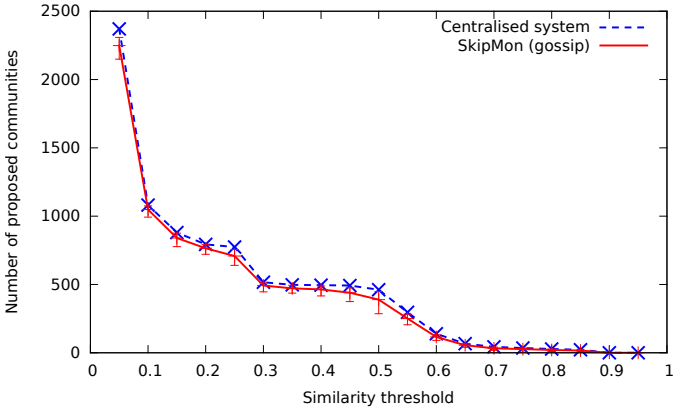

Fig. 7: Proposed communities by *SkipMon* (with gossiping) compared to a centralized system

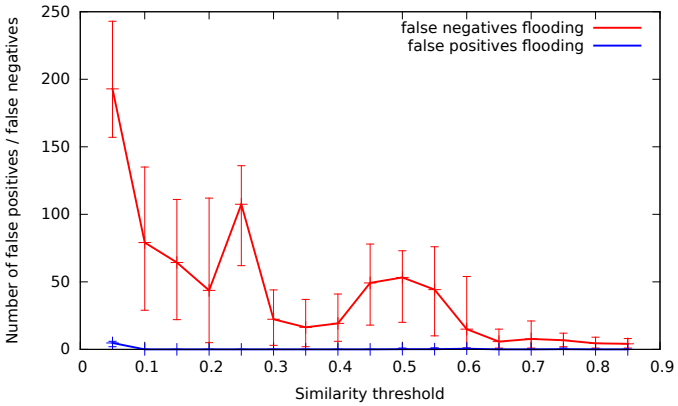

Fig. 8: False positives and false negatives (flooding)

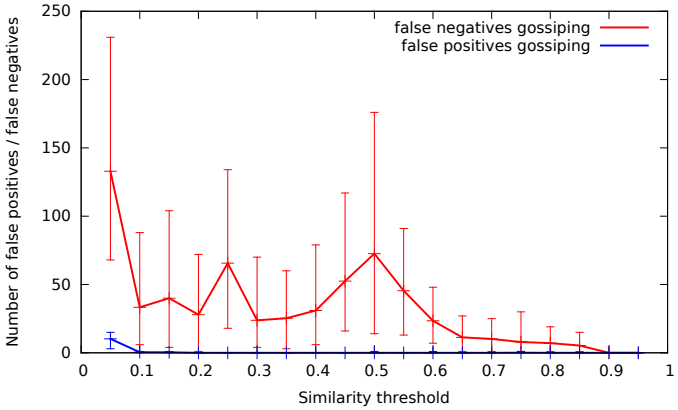

Fig. 9: False positives and false negatives (gossiping)

e.g., by dropped messages. On the other hand, the false negatives when gossiping occur due to the fact that not all nodes are communicating with each other. Moreover, with the number of total events decreasing, i.e., higher threshold, the number of false negatives also decreases. Finally, it is interesting to note that the total number of false positives is, in all cases, very low, due to the bloom filter utilization in the computation of the similarity.

2) Strict Locality: To evaluate the locality properties of *SkipMon* we create four different domains with different DNS suffixes, resulting in different name ID prefixes. As the utilized dataset itself does not provide any information regarding domains, we assign the DShield IDSs' alerts to our monitoring sensors randomly.

With respect to the locality metric this reflects to $L = 1$ for all four domains, and the dissemination mechanism in this case is partial flooding (cf. Section III-C). The results of our experiments are shown in Figure 10. As expected, this results to a much higher error compared to the centralized system, which does not follow any locality constraints. Nevertheless, in a real world scenario, we expect to have higher similarity in the alerts between nodes of the same domain and thus a higher number of proposed intra-domain communities between those nodes. Therefore, we argue that these results can be seen as the worst case scenario due to the enforced randomness in the alert creation level.

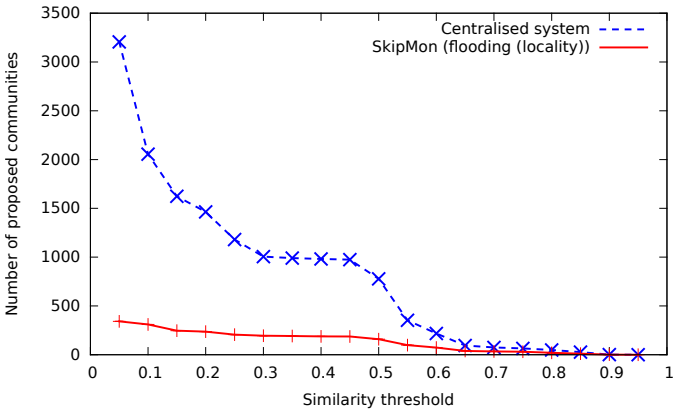

Fig. 10: Strict locality

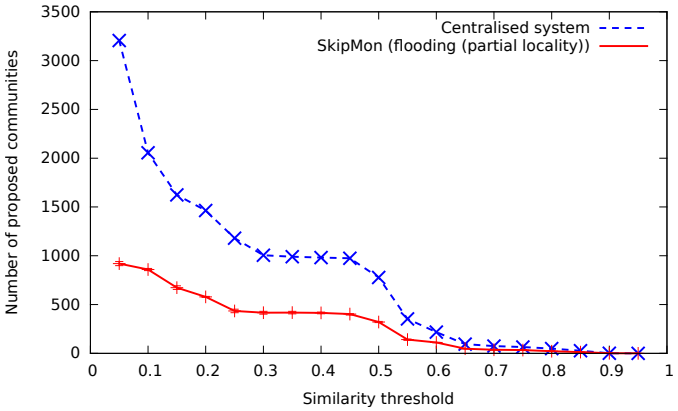

Fig. 11: Partial locality

3) Partial Locality: In order to observe a case of a possible real world scenario we examine the case of partial locality. That is, we examine how the system behaves when three of the domains keep the restrictions of internal dissemination (i.e., $L = 1$) and one of them is able to share its alerts with all nodes (i.e., $L = 0$). This scenario is depicted in Figure 11. As seen in the plot, the information shared publicly by a quarter of the nodes, enables *SkipMon* to find about twice as much communities among the nodes. Again, due to the smaller amount of messages flooded, the results of the nodes are denser over the runs.

VI. Conclusion

The number and sophistication of cyber-attacks creates a need for moving from traditional isolated IDSs to a large and distributed network of Collaborative IDSs (CIDSs). We present a novel CIDS approach that is able to distribute alerts only to monitoring sensors that are allowed to communicate with each other. Moreover, when distributing alert data the system makes sure that the privacy of such data is protected. Finally, we provide the code of our system as open-source, making it the only available for testing CIDS, while simultaneously we

give one of the first implementations and realizations of the SkipNet P2P overlay.

With regards to future work we envision the ability to exchange alert data, or summaries of alert data, between communities from multiple network domains in an hierarchical manner. Furthermore, we plan to further experiment with various data dissemination algorithms for optimizing and minimizing the communication overhead.

REFERENCES

[1] James Aspnes and Gauri Shah. Skip graphs. *ACM Transactions on Algorithms*, 3(4):37, November 2007.

[2] Frédéric Cuppens and Alexandre Miège. Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2002.

[3] Zoltán Czirkos and Gábor Hosszú. Enhancing Collaborative Intrusion Detection Methods Using a Kademlia Overlay Network. In *18th EUNICE/ IFIP WG 6.2, 6.6 International Conference*, volume 7479, pages 52–63. Springer, 2012.

[4] Herve Debar, David A. Curry, and Benjamin S. Feinstein. The Intrusion Detection Message Exchange Format (IDMEF), 2007.

[5] Vasilomanolakis Emmanouil and Krügl Matthias. SkipMon collaborative intrusion detection system. http://scm.tk.informatik.tu-darmstadt.de/projects/scm-ssi-skipmon.

[6] Nicholas JA Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. Skipnet: A scalable overlay network with practical locality properties. In *USENIX Symposium on Internet Technologies and Systems (USITS)*, volume 4, pages 1–14, Seattle, WA, 2003. USENIX Association.

[7] Ramaprabhu Janakiraman, Marcel Waldvogel, and Qi Zhang. Indra: a peer-to-peer approach to network intrusion detection and prevention. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE)*, pages 226–231. IEEE, 2003.

[8] Anne Marie Kermarrec, Laurent Massoulié, and Ayalvadi J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):248–258, 2003.

[9] Anne-Marie Kermarrec and Maarten van Steen. Gossiping in distributed systems. *ACM SIGOPS Operating Systems Review*, 41(5):2, 2007.

[10] Michael E. Locasto, Janak J. Parekh, Angelos D. Keromytis, and Salvatore J. Stolfo. Towards Collaborative Security and P2P Intrusion Detection. In *IEEE Workshop on Information Assurance and Security*, pages 333 – 339. IEEE, 2005.

[11] EK Lua, J Crowcroft, and M Pias. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, 7(2):72–93, 2005.

[12] Mirco Marchetti, Michele Messori, and Michele Colajanni. Peer-to-Peer Architecture for Collaborative Intrusion and Malware Detection on a Large Scale. *Lecture Notes in Computer Science*, 5735:475–490, 2009.

[13] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *Peer-to-Peer Systems*, volume 2429 of *LNCS*. Springer Berlin Heidelberg, 2002.

[14] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[15] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, December 1999.

[16] Phillip A. Porras and Peter G. Neumann. EMERALD: Event monitoring enabling response to anomalous live disturbances. In *National information systems security conference (NISSC)*, pages 353–365, 1997.

[17] William Pugh. Skip lists: a probabilistic alternative to balanced trees. *Communications of the ACM*, 33(6):668–676, 1990.

[18] Martin Roesch. Snort-lightweight intrusion detection for networks. In *USENIX conference on System administration*, pages 229–238, 1999.

[19] Steven Snapp, James Brentano, Gihan Dias, Terrance Goan, Todd Heberlein, Che-Lin Ho, Karl Levitt, Biswanath Mukherjee, Stephen Smaha, Tim Grance, Daniel Teal, and Doug Mansur. DIDS (Distributed intrusion detection system) - Motivation, Architecture, and an early Prototype. In *Fourteenth National Computer Security Conference*, pages 167–176, 1991.

[20] Aditya K. Sood and Richard J. Enbody. Targeted Cyber Attacks-A Superset of Advanced Persistent Threats. *IEEE Security & Privacy*, 11(1):54–61, 2013.

[21] Eugene H Spafford and Diego Zamboni. Intrusion Detection using Autonomous Agents. *Computer Networks*, 34(4):547–570, 2000.

[22] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan Ý. Chord : A Scalable Peer-to-peer Lookup Service for Internet. In *Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM, 2001.

[23] Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and Practice of Bloom Filters for Distributed Systems. *IEEE Communications Surveys & Tutorials*, 14(1):131–155, 2012.

[24] Johannes Ullrich. Dshield internet storm center. https://www.dshield.org/, 2000.

[25] Emmanouil Vasilomanolakis, Shankar Karuppayah, Panayotis Kikiras, and Max Mühlhäuser. A honeypot-driven cyber incident monitor: lessons learned and steps ahead. In *International Conference on Security of Information and Networks*. ACM, 2015.

[26] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. Taxonomy and Survey of Collaborative Intrusion Detection. *ACM Computing Surveys*, 47(4):33, 2015.

[27] Zheng Zhang, Jun Li, C N Manikopoulos, Jay Jorgenson, and Jose Ucles. HIDE : a Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. In *IEEE Workshop on Information Assurance and Security*, pages 85–90. IEEE, 2001.

[28] Chenfeng Vincent Zhou, Shanika Karunasekera, and Christopher Leckie. A peer-to-peer collaborative intrusion detection system. In *International Conference on Networks*, pages 118–123. IEEE, 2005.

[29] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. A Survey of Coordinated Attacks and Collaborative Intrusion Detection. *Computers & Security*, 29(1):124–140, February 2010.