# SMURFEN: A System Framework for Rule Sharing Collaborative Intrusion Detection

Carol Fung[1], Quanyan Zhu[2], Raouf Boutaba[1, 3], and Tamer Başar[2]
[1]David R. Cheriton School of Computer Science,
University of Waterloo, Ontario, Canada, {j22fung,rboutaba}@uwaterloo.ca
[2]Department of Electrical and Computer Engineering,
University of Illinois at Urbana-Champaign, USA. {zhu31,basar1}@illinois.edu
[3]Division of IT Convergence Engineering, POSTECH, Pohang, KB 790-784, Korea.

***Abstract*—Intrusion Detection Systems (IDSs) are designed to monitor network traffic and computer activities in order to alert users about suspicious intrusions. Collaboration among IDSs allows users to benefit from the collective knowledge and information from their collaborators and achieve more accurate intrusion detection. However, most existing collaborative intrusion detection networks rely on the exchange of intrusion data which raises privacy concerns. To overcome this problem, we propose SMURFEN: a Rule Sharing intrusion detection network, which provides a platform for IDS users to effectively share their customized detection knowledge in an IDS community. An automatic rule propagation mechanism is proposed based on a decentralized two-level optimization problem formulation. We evaluate our rule sharing system through simulations and compare our results to existing knowledge sharing methods such as random gossiping and fixed neighbors sharing schemes.**

## I. Introduction

In recent years, Internet intrusions have become more sophisticated and difficult to detect. With the increasing complexity of software and systems, thousands of vulnerabilities are being discovered and exposed for exploitation every year. Attacks usually appear before security vendors release their defense technology and software vendors release their corresponding patches (e.g., zero-day attacks). Attacks from the Internet are usually accomplished with the assistance of malicious code (a.k.a. malware), including worms, viruses, Trojan horses, and Spyware. An example is the Conflicker worm which infected more than 3 million servers from year 2008 to 2009, with an estimated economic loss of $9.1 billion [1]. Recent intrusion attacks compromise a large number of nodes to form botnets. Hackers not only harvest private data and identify information from compromised nodes, but also use those compromised nodes to launch attacks such as distributed-denial-of-service (DDoS) attacks, distribution of spam messages, or organized phishing attacks.

To protect computer users from malicious intrusions, Intrusion Detection Systems (IDSs) are designed to monitor network traffic and computer activities by raising intrusion alerts to network administrators or security officers. IDSs can be categorized into host-based (HIDS) or network-based (NIDS) according to their targets, and signature-based or anomaly-based according to their detection methodologies. A NIDS monitors the network traffic from/to one or a group of computers and compare the data with known intrusion patterns. A HIDS monitors the activities of one computer but has a deeper insight by tracking the file systems and system logs of the host computer. A signature-based IDS identifies malicious code if a match is found with a pattern in the attack signature database. An anomaly-based IDS, on the other hand, monitors the traffic volume or behavior of the computer and raise alerts when they are out of a predefined normal scope. Compared to HIDS, an NIDS has a broader view of the status of the network it monitors, but may miss some intrusions which are hard to detect by observing network traffic only. A signature-based IDS can accurately identify intrusions and the false positive rate is low compared to anomaly-based detection. However, it is not effective for zero-day attacks, polymorphic, and metamorphic malware [2]. An anomaly-based IDS may detect zero-day attacks by analyzing their abnormal behaviors. However, an anomaly-based detection usually generates a high false positive rate.

Traditional IDSs work independently from each other and rely on downloading new signatures or detection rules from the corresponding security vendor's signature/rule base to remain synchronized with new detection knowledge. However, the increasing number and diversity of intrusions render it not effective to rely on the detection knowledge from a single vendor, since not a single vendor can cover all the possible intrusions due to limited labor and available technology. Indeed, vendors usually choose to cover high priority intrusions which may have large influence among their clients or have high risk levels. Collaborative intrusion detection networks (CIDNs) provide a platform for IDSs to take advantage of the collective knowledge from collaborators to improve the overall detection capability and accuracy. However, most existing CIDNs, such as [3], [4], [5], [6], and [7], rely on the sharing of intrusion data with others, which raises privacy concerns. Instead, sharing detection knowledge such as malware signatures and intrusion detection rules, causes less privacy concern.

In reality, expert IDS users, including security analysts, network administrators, and security system programmers,

The research by the authors from University of Illinois is partially supported by AFOSR under MURI Grant FA9550-09-1-0643 and by the Boeing Company through Information Trust Institute. This research is also partially supported by the Natural Science and Engineering Council of Canada (NSERC) under its discovery program and partially by WCU (World Class University) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

create their own detection rules or customize existing ones to improve detection accuracy specifically for their individual environment [8]. A new detection rule created by one user may be adopted directly by another user if they have similar network/computer configurations. For example, a new intrusion detection rule created to minimize vulnerability of a software can be adopted by others using the same software. An expert user who creates new rules for newly revealed vulnerabilities may share their rules with others who are subject to similar vulnerabilities and interests. Sharing rules among a large group of users can be an effective way to improve the overall security among all users.

In this paper, we leverage the benefit of intrusion detection knowledge sharing and propose SMURFEN, a rule sharing collaborative intrusion detection network, where intrusion detection knowledge is shared among users who share similar interests in the community. The framework is based on a peer-to-peer overlay, where each user maintains a list of collaborators and send his/her feedback through the P2P system. Accordingly, an automatic knowledge dissemination mechanism is proposed to allow users effectively share detection rules with other users without overwhelming their receiving capacities. We demonstrate using simulation that the proposed rule sharing mechanism can effectively improve the overall security of the community and provides incentive-compatibility and fairness to the collaborators.

The rest of the paper is organized as follows. In Section II, we give an overview of collaborative intrusion detection systems and information sharing paradigms. We evaluate the proposed system using simulation in Section IV. Finally, we conclude the paper in Section V.

## II. Related Work

Traditional IDS collaboration utilizes the collective intrusion information and knowledge from other IDSs to improve accuracy in intrusion detection. Existing CIDNs can be categorized into information-based and expertise-based. In an information-based CIDN, IDSs collect intrusion data such as intrusion alerts or firewall logs from other nodes to perform overall intrusion detection for the whole network. Most works proposed in the last few years are information-based CIDNs, such as [3], [4], and [5]. They are especially effective in detecting epidemic worms or attacks, and most of them require homogeneous participant IDSs. In an expertise-based CIDN, suspicious data samples are sent to expert collaborators for diagnosis. Feedbacks from the collaborators are then aggregated to help the sender IDS detect intrusions. Examples of such CIDNs include those given in [9] and [7]. Expertise-based CIDNs may involve heterogeneous IDSs and are effective in detecting many intrusion types including malware, scannings, and vulnerability exploitations.

However, both types of CIDNs rely on the sharing of intrusion data, which raises privacy concerns. Therefore, it greatly discourages users from collaborating with unknown parties. In contrast, sharing detection knowledge, such as detection rules and malware signatures, does not involve the sharing of sensitive data. Hence, it can effectively eliminate the privacy concern in IDS collaboration. In fact, some open source IDSs, such as Snort [10], use mailing lists to allow users to contribute and share their own detection rules. However, mailing lists do not provide customized filtering and they do not scale well either, making it inefficient for frequent knowledge exchange within large communities.

Information and knowledge propagation in a community can be realized through gossiping. Gossiping is a communication paradigm where information is propagated through multi-hop pair-wise communication. Gossiping has been used to exchange information in distributed collaborative intrusion detection, such as local gossiping [11], and global gossiping [12]. Sharing observations from distributed nodes is useful to detect and throttle fast spreading computer worms. It is effective for communications in ad hoc or random networks, where a structured communication topology is hard to establish. However, traditional gossiping relies on random pairs-wise communication and information flooding. Therefore, it is not suitable when the network is large and the messages are only intended to be delivered to a small set of nodes. Mailing list broadcasting can be seen as a special type of gossiping where one node communicates with every other node in the network to deliver messages. Most existing IDS vendors, such as Snort, use broadcasting to deliver their vendor certified intrusion detection rules.

Publish-subscribe systems can also be used for information delivery among IDSs, such as [13], [5]. Compared to gossiping, publish-subscribe systems allow customized information delivery. They can be either topic-based [14], or content-based, such as [13], [5]. In a topic-based system, publishers and subscribers are connected by predefined topics; content is published on well-advertised topics to which users can subscribe based on their interests. In a content-based system, users' interests are expressed through queries, and a content filtering technique is used to match the publishers' content to the subscriber. However, a simple publish-subscribe system does not take into account the quality of the information. It also does not provide incentives for IDSs to contribute to the collaboration network. SMURFEN measures the trust of nodes, and provides an incentive-compatible rule sharing.

## III. SMURFEN: A Detection Rule Sharing Intrusion Detection Network

An intrusion detection rule is a detection policy which specifies the pattern of suspicious attacks. Each rule can trigger an alert once the pattern is matched. Detection rules can be vulnerability-based or exploit-based. A vulnerability is a software defect or system misconfiguration that allows attackers to gain access or interfere with system operations. Common examples of software vulnerabilities are software buffer overflows and HTTP header injection. A vulnerability-based detection rule specifies the pattern of attacks on a specific vulnerability. The patterns can be the IP address, port number, protocol flags, and context of the data payload. An exploit-based detection rule specifies the common patterns

of general attacks. Comparatively the exploit-based detection causes higher false positives than vulnerability-based detection, but is effective when the vulnerabilities are unknown.

Defense against attackers is a challenging problem since a defender needs to know all possible attacks to ensure network security, whereas an attacker only needs to know a few attack techniques to succeed. It is often impossible for one person or a small group of defenders to know all attack techniques but is common to have knowledge about some attacks. As a result, the attackers have a significant advantage over the defenders. This motivates defenders to share knowledge with others to overcome their weakness. In fact, some open source intrusion detection systems, such as Snort and OSSEC [15], allow users to create and edit detection rules, which provides an opportunity for users to contribute and exchange intrusion detection rules. The purpose of SMURFEN is to provide such a platform for users to share their detection rules with others effectively. We focus on an efficient rule sharing mechanism design and compare it with other possible solutions such as random gossiping and fixed neighbors sharing mechanisms.

### A. The SMURFEN Framework

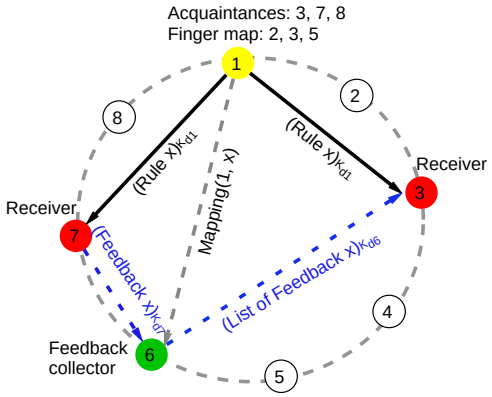


Fig. 1. SMURFEN design of 8 nodes on a Chord ring: nodes 3 and 7 receive a rule from node 1. The feedbacks are collected by node 6.

The SMURFEN framework is built on a DHT-based peer-to-peer (p2p)[16] communication overlay as illustrated in Fig. 1. Each node maintains a list of neighbors to communicate and exchange intrusion detection rules with. We call such a list the *acquaintance list*. In the rest of this paper, we use the terms *acquaintance* and *neighbor* interchangeably. Note that the acquaintance relationship is symmetric, i.e., if node $i$ is in node $j$'s acquaintance list, then node $j$ is in node $i$'s acquaintance list. Each node may have a long list of acquaintances and each acquaintance $j$ has a certain probability $p_{ij} \in (0, 1]$ to be chosen to receive rules from the sender node $i$. A user on the receiver side evaluates rules sent from its neighbors and may choose to "*accept*" or "*reject*" the rule. The decision is then recorded by a Bayesian learning algorithm to update the *trust value* of the sender. The *trust value* from $i$ to $j$ is the probability that the rules from the sender $i$ are useful to the receiver $j$. The higher a collaborator's trust, the more helpful it is in collaboration. The decision is also sent to a corresponding rule feedback collector. The feedback collector is a random node in the P2P network, determined by a hashed key from the rule ID and the sender ID. The corresponding node holding the key will host the feedback of the rule. Inexperienced users can check feedback from others before they make their own decision whether to accept the rule or not. Users can also report false positives and true positives about the rule, so that the rule creator can collect feedback and make updates accordingly. More details about the feedback collector are provided in section III-E.

### B. Snort Rules

Many intrusion detection systems, such as Snort, allow users to create and edit their own detection rules in their rule base. Snort base rules are certified by the Vulnerability Research Team (VRT) [17], after being tested by security experts. Snort rules are vulnerability-based and written in plain text; hence can be easily interpreted and edited by users. Snort rules obtained from third parties can be adopted directly or indirectly with some changes. Snort rules can be independent or can be grouped together into rule units. The basic rule structure includes two logical sections: the header section and the option section. The rule header contains the rule's action, protocol, source and destination IP addresses and network masks, and the source and destination ports information. The rule option section contains alert messages and information on which parts of the packet should be inspected to determine whether the rule action should be taken [18]. Fig. 2 illustrates a simple Snort rule. When a TCP packet with the destination IP and port number matching the specified pattern and data payload containing the specified binary content is detected, a "mounted access" alert is raised.

```
alert tcp any any -> 192.168.1.0/24 111 \
    (content:"|00 01 86 a5|"; msg:"mountd access";)
```

Fig. 2. An Example of Snort Rule (adapted from [18])

### C. Join or Leave SMURFEN

To prevent the man-in-the-middle attack, the communication between each pair of nodes is signed by the private key of the sender. When a new node joins the network, it creates a public/private key pair $(K_e, K_d)$, and registers a new ID into the p2p network by sending a join request to any node in the network. After that, the new node sends connection requests to random nodes in the network and acquaintance relationships are established when the requests are accepted. When a node leaves a network, it is not required to send a notification to other nodes. When a node does not receive response from an acquaintance, it automatically sets the acquaintance status to be inactive and seeks new replacement.

### D. Trust Evaluation and Acquaintance Management

Each node in the network shares its intrusion detection knowledge with their acquaintances. However, trust evaluation is necessary to distinguish good/bad nodes in the network. Each IDS evaluates the trust values of others by rating the

quality of the intrusion detection knowledge from them. If a detection rule is accepted, a corresponding credit is recorded for the sender. If a rule is rejected, a debit is recorded for the sender. A Bayesian learning algorithm [19] is then used to update the *trust value* of the sender based on the usefulness of rules sent from the past. An accept will increase the trust value of the sender and a reject will penalize it. The *trust value* of node $i$ perceived by node $j$ can be seen as the level of helpfulness that node $i$ is to the receiver node $j$. The more helpful a collaborator, the higher its trust value.

The acquaintance relationship is based on a mutual consent. Every new acquaintance is assigned a low trust value at the beginning and needs to pass a probation period before becoming a collaborator. During the probation period, the trust value of the new acquaintance will be evaluated by its peers. When the probation period expires, new acquaintance gaining high trust values will replace those with low trust values in the list. Acquaintances with trust lower than a certain threshold will be removed and new ones are recruited regularly.

### E. Feedback Collector

When a user receives new rules from the community, she/he may evaluate the rules and determine whether or not to adopt the rule. A SMURFEN system includes feedback collectors to record the feedback on the rules from users. Less experienced users may check the feedback from others before making their decisions. As shown in Fig. 3, rule author "$A$" propagates a new rule $i$ to its acquaintances $R_1$ and $R_2$. Both rule receivers can retrieve and send feedback from/to the feedback collector $C$, which is a random node in the P2P network determined by the key mapping of the creator and the rule ID. Replicas collectors can be used to improve the availability of feedback collector service. All feedbacks are signed by their authors to prevent from malicious tampering.

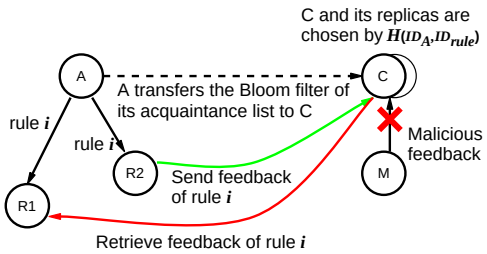


Fig. 3. Feedback Collection in SMURFEN. The malicious node $M$ attempts to leave fraudulent feedback but was blocked since it does not match the Bloom filter on the feedback collector.

Moreover, to avoid feedback fraudulence, each feedback collector maintains a Bloom filter [20] of the authorized nodes list. The rule author hashes all of its acquaintances into a Bloom filter and passes it to the feedback collector. Only nodes with hashed IDs matching the Bloom filter are allowed to leave feedback on the collector. The use of Bloom filter not only reduces the communication overhead to transfer long acquaintance lists, it also avoids unnecessary information leaking from the rule author.

### F. SMURFEN Knowledge Propagation

Intrusion detection knowledge propagation mechanism is an essential part of the SMURFEN system, where IDSs decide the propagation rates to their neighbors. An appropriate propagation design will not only provide incentive-compatibility which discourages free-riders and rewards contributors, it will also provide fairness to all participants and be robust to malicious insiders. In this work, we use a game-theoretical approach for each IDS to decide its rule propagation rates and we prove that the system yields to a Nash equilibrium.

We model our system based on a two-level optimization problem, i.e., a *public* utility optimization together with a *private* utility optimization. Each IDS $i$ controls two decision variables, namely, $\vec{r}_i$ and $\vec{R}_i$. $\vec{r}_i$ is the *rule propagation rate* from node $i$ to its neighbors. To prevent from denial of service attacks from malicious neighbors, a node $i$ also sets a *requested sending rate* $\vec{R}_i$, which sets the upper bound of the sending rates from all neighbors. At the lower level, a node $i$ solves the public optimization problem (PP$i$) where it chooses $\vec{r}_i$ to maximize the aggregated satisfaction levels of its neighbors. The public objective function aggregates the satisfaction level of all neighbors by the their trust factors. The public utility can be viewed as a public altruistic utility in that a node $i$ seeks to satisfy its neighbors by choosing rule propagation rates $\vec{r}_i$. At the upper level, a node $i$ determines $\vec{R}_i$ to solve a private optimization problem (P$i$) to maximize the total return benefit from all neighbors. The return benefit is the aggregated return from neighbors weighted by their trust. The private objective indicates that a node intends to maximize its own level of satisfaction by choosing an appropriate level of request. The request capacity is imposed to prevent excessive incoming traffic as a result of high level of requests. The choice of $\vec{R}_i$ at the upper level influences the decision-making at the lower optimization level.

The entailed mathematical description of the model for rule propagation can be found in [21]. We have shown that the knowledge propagation model based on the two-level design possesses a Nash equilibrium that satisfies the property $r^*_{ij} = R^*_{ij}, \forall i, j \in \mathcal{N}$, which we call a *prime* Nash equilibrium [21].

### G. An Example

For a better understanding of the rule sharing framework, we illustrate the mechanism with an example (see Fig. 1). Assume that user 1 (on node 1) detects a new software vulnerability and creates a new Snort rule $x$ to protect the system before the official release from the VRT. User 1 is part of the rule sharing network. The new rule is automatically propagated to its acquaintances through a propagation process (to be described in Section III-F). User 3 and user 7 receive rule $x$ from user 1. The user 7 finds rule $x$ to be useful to her/his network and can choose to *accept* or *reject* it. The decision is then notified to a feedback collector on node 6. If the rule is adopted and alerts are triggered by rule $x$, the decision whether it is a true or false alarm is also forwarded to node 6. Users can reject a formally accepted rule any time when it causes large false positives or does not detect any attack

after a certain amount of time. Rule $x$ is also propagated to node 3. The user 3 finds that the rule covers vulnerabilities but does not have enough experience to judge the quality of the rule, she/he chooses to inspect the feedback from other users about the rule from the feedback collector. The decision of acceptance or rejection can be delayed to allow enough time for observation.

### H. Discussion

Our rule propagation is based on the assumption that users of the IDSs are capable of understanding the exchanged rules and make "accept" or "reject" decisions based on their judgments. Inexperienced users can take advantage of the feedback collector system to see the ranking of the new rule before making their decisions. Note that duplicated rules will be automatically filtered out in the receiver side. For example, user receives rule #$x$ from both neighbor $A$ and neighbor $B$. The same rule from neighbor $B$ will be abandoned since it came later than the other one. Users should also have the capability to distinguish detection rules which have overlapping functionalities. For example, different rules targeting on the same threat may not be adopted by the receiver since the receiver notice the overlap. This way, only the neighbor sent the earliest version of detection rules will be rewarded.

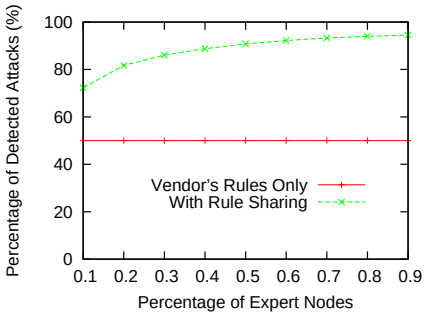


Fig. 4. Intrusion Detection Accuracy with and without Rule Sharing

## IV. Evaluation

In this section, we use a simulation network to demonstrate the appealing properties of the SMURFEN system. All our experiments are based on the average of a large number of experiment replications with different random seeds. Confidence intervals are small enough to be neglected.

### A. Simulation Setup

We simulate a network of $n$ nodes. Each node $i \in \{1, 2, \cdots, n\}$ is labeled with an expertise level $e_i \in [0, 1], \forall j \in \mathcal{N}$, which is the probability that a rule propagated by node $i$ is effective for intrusion detection. Note that the higher the expertise level, the higher the trust value. Each node $i$ contributes detection rules to the network following a Poisson distribution with an average arrival rate $\widehat{r}_i$. $T_{ij}$ is learned by $j$ through past experiences using the Bayesian learning method described in [19]. The rule propagation follows the two-level game design described in Section III-F. In this section, we show some selected results on efficiency, incentive compatibility, fairness, and robustness of the system.

### B. Intrusion Detection Accuracy

In this experiment, we evaluate the effectiveness of intrusion detection using rule sharing. We configure a network of $100$ nodes with the same set of vulnerabilities, comprising expert nodes with expertise level $0.9$ and novice nodes with expertise level $0.1$. Each node has on average $10$ randomly selected neighbors. We simulate $20$ attacks on the network. $10$ attacks are detectable by vendor released rules, and $10$ attacks are not covered by the vendor but detectable by rules created and shared among the IDSs in the network. We observe the average percentage of attacks detected by each IDS in the network, with and without rule sharing, and with different ratios of expert nodes. Fig. 4 shows that with the rule sharing capability, the average percentage of detected attacks is improved significantly compared to the case without sharing. The higher the ratio of expert nodes, the higher the detection rate. This is because expert nodes effectively improve the propagation of high quality rules in the network.

### C. Information Quality

In this experiment we compare the information quality using the traditional mailing list and SMURFEN propagation system. When using mailing list propagation method, detection rules are broadcast to all users in the network, while in SMURFEN rules are propagated following the two-level optimization game. We set up a network with size starting from $10$ nodes and we increase it by $30$ nodes each round till $130$. Among all the nodes, $20\%$ are expert nodes with expertise level $0.9$, $80\%$ are novice nodes with expertise level $0.1$.

Fig. 5 shows the information quality for both methods. We define the information quality to be the percentage of useful rules that nodes receive. We see that when using the SMURFEN system, the information qualities received by both the low-expertise and the high-expertise nodes are significantly improved compared to the mailing list method. The high-expertise nodes receive higher quality rules than low-expertise nodes, which reflects the incentive-compatibility of the system.

### D. Incentive Compatibility and Fairness

Incentive compatibility is a required feature for a collaboration network since it determines the long-term sustainability of the system. In an incentive-compatible system, a well-behaving node benefits more than an ill-behaving one. In this experiment, we vary the expertise level of a participating node, and observe the output of its return benefit, which is the expected number of useful rules a node receives per day.

In this experiment, we configure a network with $30$ nodes with random expertise levels in $[0, 1]$. We change the expertise level of node $0$ from $0.1$ to $1.0$ and observe its return benefit. We compare our results with two other information propagation methods, namely uniform gossiping and best neighbor mechanism. In the uniform gossiping mechanism, rules are propagated to randomly selected nodes uniformly from the neighborhood. The receiver drops rules from less compatible neighbors if the total receiving rate is over limit. In the best neighbor mechanism, rules are always propagated to a few fixed (best) neighbors. The sending capacity and receiving

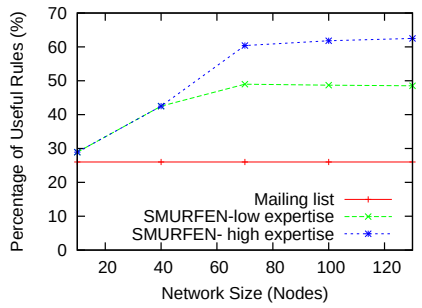


Fig. 5. Information Quality Comparison

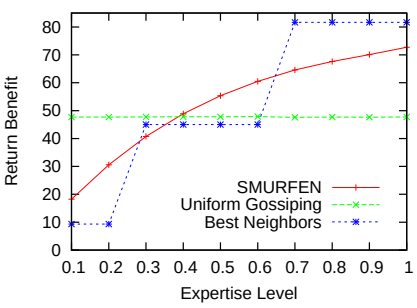


Fig. 6. Incentive on Expertise Levels

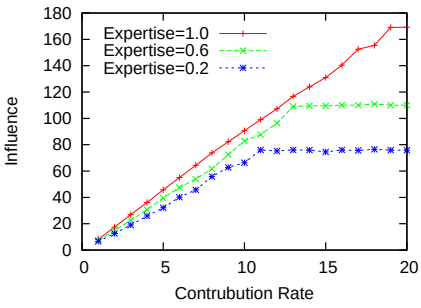


Fig. 7. The Influence vs. Sending Rate

capacity also apply to the uniform gossiping and best neighbor propagation. Therefore, we also configure their sending and receiving capacities to be 100 rules/day.

Fig. 6 shows that uniform gossiping provides no incentive to nodes with higher expertise levels. On the other hand, the best neighbor propagation scheme provides incentive but no fairness. Nodes of the same expertise levels may have very different return benefit. This is because under the best neighbor mechanism, nodes form collaboration groups. Nodes of the same expertise level may join different groups. Since the return benefit largely depends on which group a node belongs to, nodes with the same expertise levels may have significantly different return benefit. On the contrary, SMURFEN has a continuous concave utility on the return benefit over expertise levels. It ensures incentive compatibility as well as fairness.

### *E. Robustness of the System*

The purpose of this experiment is to demonstrate the robustness of the system in the face of denial-of-service attacks. We define the influence of a node $i$ to be the total number of rules received by all the neighbors of $i$ per day. The larger the influence of a node, the higher potential of damage the node can cause once it is malicious. From Fig. 7, we see that the influence of a node is bounded in the system. This is because the SMURFEN system enforces propagation agreements between each pair of nodes. Each node sets a propagation limit to all its neighbors using the two-level game (see Section III-F). Therefore, when a node intends to launch a DoS attack, the amount of rules it is allowed to send to others is bounded by the limits set by its neighbors. Nodes sending excessive traffic to neighbors will be revealed as potential malicious nodes, and thus removed from the neighbor list of others.

## V. Conclusion

In this paper, we have introduced a peer-to-peer rule sharing framework called SMURFEN for collaborative intrusion detection. The propagation mechanism has been derived from a decentralized two-level optimization problem formulation. We have shown that our system effectively improves the system-wide intrusion detection accuracy, and has the properties of incentive compatibility, fairness, scalability, and robustness to denial-of-service attacks. By simulation, we have corroborated these important CIDN properties. As future work, we intend to show system robustness to different insider attacks and design a resilient and robust topology for the collaborative network.

## References


[1] "ZDnet." http://www.zdnet.com/blog/security/conficker-estimated-economic-cost-91-billion/3207 [Last accessed in July 6, 2011].

[2] P. Li, M. Salour, and X. Su, "A survey of internet worm detection and containment," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 1, pp. 20–35, 2008.

[3] J. Ullrich, "DShield." http://www.dshield.org/indexd.html.

[4] V. Yegneswaran, P. Barford, and S. Jha, "Global intrusion detection in the domino overlay system," in *NDSS'04*.

[5] M. Cai, K. Hwang, Y. Kwok, S. Song, and Y. Chen, "Collaborative internet worm containment," *IEEE Security & Privacy*, vol. 3, no. 3, pp. 25–33, 2005.

[6] C. Fung, O. Baysal, J. Zhang, I. Aib, and R. Boutaba, "Trust management for host-based collaborative intrusion detection," in *19th IFIP/IEEE Intl. Workshop on Distributed Systems*, 2008.

[7] J. Oberheide, E. Cooke, and F. Jahanian, "Cloudav: N-version antivirus in the network cloud," in *Proc. of the 17th USENIX Security Symp.*, 2008.

[8] J. Goodall, W. Lutters, and A. Komlodi, "I know my network: collaboration and expertise in intrusion detection," in *ACM conf. on Computer supported cooperative work*, 2004.

[9] C. Fung, J. Zhang, I. Aib, and R. Boutaba, "Robust and scalable trust management for collaborative intrusion detection," in *11th IFIP/IEEE Intl. Symp. on Integrated Network Management*, 2009.

[10] "Snort." http://www.snort.org/ [Last accessed in July 6, 2011].

[11] D. Dash, B. Kveton, J. Agosta, E. Schooler, J. Chandrashekar, A. Bachrach, and A. Newman, "When gossip is good: Distributed probabilistic inference for detection of slow network intrusions," in *AAAI'06*.

[12] G. Zhang and M. Parashar, "Cooperative detection and protection against network attacks using decentralized information sharing," *Cluster Computing*, vol. 13, no. 1, pp. 67–86, 2010.

[13] C. Zhou, S. Karunasekera, and C. Leckie, "Evaluation of a decentralized architecture for large scale collaborative intrusion detection," in *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE Intl. Symp. on*, pp. 80–89, IEEE, 2007.

[14] V. Ramasubramanian, R. Peterson, and E. G. Sirer, "Corona: a high performance publish-subscribe system for the world wide web," in *NSDI'06*.

[15] "OSSEC." http://www.ossec.net/ [Last accessed in July 6, 2011].

[16] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *ACM SIGCOMM*, pp. 149–160, ACM, 2001.

[17] http://www.csoonline.com/article/593237/inside-sourcefire-s-vulnerability-research-team?page=2 [Last accessed in July 6, 2011].

[18] M. Roesch and C. Green, "Snort users manual," *Snort Release*, vol. 1, April 2010.

[19] C. Fung, Q. Zhu, R. Boutaba, and T. Başar, "SMURFEN: A Knowledge Sharing Intrusion Detection Network," Tech. Rep. CS-2011-06, University of Waterloo, http://www.cs.uwaterloo.ca/research/tr/2011/CS-2011-06pdf.pdf, 2011.

[20] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[21] Q. Zhu, F. Fung, R. Boutaba, and T. Başar, "A game-theoretic approach to rule sharing mechanism in networked intrusion detection systems," in *Proc. of 50th IEEE CDC and ECC, To appear*, 2011.