QOI: ASSESSING PARTICIPATION IN THREAT INFORMATION SHARING

Jeman Park[1], *Hisham Alasmary*[1], *Omar Al-Ibrahim*[1]
Charles Kamhoua[2], Kevin Kwiat[3], Laurent Njilla[3], and Aziz Mohaisen[1]

[1]University of Central Florida, [2]Army Research Laboratory, [3]Air Force Research Laboratory

ABSTRACT

We introduce the notion of Quality of Indicator (QoI) to assess the level of contribution by participants in threat intelligence sharing. We exemplify QoI by metrics of the correctness, relevance, utility, and uniqueness of indicators. We build a system that extrapolates the metrics using a machine learning process over a reference set of indicators. We compared these results against a model that only considers the volume of information as a metric for contribution, and unveiled various observations, including the ability to spot low-quality contributions that are synonymous to free-riding.

Index Terms— Threat intelligence, sharing, QoI

1. INTRODUCTION

In threat intelligence sharing, participants exchange patterns of threats, in the form of threat indicators or signals, with each other [16, 4, 25, 24, 8, 13]. Participants are defined over a community of trust and ideally collaborate towards a common mission: to understand and respond to emerging threats [28]. For such intelligence sharing to happen, standards for representation, exchange, and consumption of indicators are used [3, 11, 14, 21, 30, 10, 5]. Communities of trust are established, and systems and initiatives for sharing are built. However, participants need to contribute information in those systems to be consumed by other community members. Given the coinciding benefits and risks of threat information sharing, some community members have adopted an elusive behavior of "free-riding" [6] so that they can achieve utility of the sharing paradigms without contributing much to the community.

Understanding the effectiveness of sharing has been viewed from the point of view of whether participants contribute or not, and using the volume of contributed indicators. Therefore, a community member who does not contribute a volume of data is considered a free-riding community member [27]. The state-of-the-art on the problem did not include other metrics beyond simple measures of volume-based contribution, particularly metrics that evaluate qualitatively the indicator, which means the threat information contributed by participants. To the best of our knowledge, while sparsely mentioned in other work [28, 29], this problem is not treated properly in the literature. Thus, this work is the first dedicated to the problem by identifying the Quality of Indicator (QoI) as a new metric of contribution to capture contribution in information sharing for threat intelligence. An ideal measure of QoI should distinguish between the various members based on their contribution. With the possible specialty of community members and the varying uses of indicators shared based on the context in which they are used, a major challenge is to assign context-dependent quality markers for indicators.

The correctness, relevance, utility, and uniqueness are yet a few other quality measures that we consider in this work. The *correctness* captures the accuracy of annotation and labeling of an indicator, which is important in designing the proper countermeasures and defenses to attacks and threat. The *relevance* of an indicator to the community members captures how aligned a given contributed indicator with a community member's needs. The *utility* captures whether an indicator characterizes prominent features of cyber-threats. Finally, the *uniqueness* is defined as a measure of similarity with previously seen indicators. Our method of QoI is based on exploiting a fine-grained record as a benchmark for assessing contributions of community members.

2. QOI ASSESSMENT

2.1. System Architecture and Design

Strawman Design. Before nodes can accept indicators, they first evaluate their quality by asking a special node, an assessor, to perform rating of the indicators. While effective, this strawman design has multiple issues. Most importantly, each community member has to fully trust the assessor and the validity of its scoring of indicators. Second, the approach is prone to disruption by the failure of the assessor. Such issues could be, however, mitigated by introducing multiple assessors, where the final score of quality is obtained using a consensus over multiple scores, one per assessor.

Distributed Design. On the other end of the spectrum of the strawman design is to let every node in the system to be an assessor. In doing that, we implicitly assume the availability of reference data to each node in the system, which is implausible. For example, based on a prior study, no single community member (antivirus scanner in the case of malware detection and labeling) has a 100% coverage or accuracy [17]: namely, it takes between 6 and 18 community members to provide close to perfect coverage of detection and correctness of labeling, respectively, for a malware family like Zeus [18]. This, in turn, calls for a more "intelligent" process for the assessment of QoI. Our system assumes the reference dataset has sufficient information about every indicator presented by the different community members. A high-level description of the system implementation, incorporating both learning and assessment, is shown in Figure 1. Note that this system is ideally executed by each community member.

2.2. System Setup

We use malware indicators sharing as a working example. Among the steps described below, each process from S1 to S3 is matched with box 1 through 3 of Figure 1.

◇ S0: **Defining Metrics.** Quality metrics are used to ensure that community members who participate in information sharing provide threat indicators that are valuable to other members, while scoring procedures are methods that specify how these metrics are used to generate a quality score. The main purpose of this work is defining those metrics.

◇ S1: **Defining Labels.** Annotations capture the type of threat, the level (of severity, timeliness, etc.) or quality type of an indicator. Utilizing these annotations, a weight value is assigned to each quality label, and a scoring method is used to convert the quality labels to a numeric aggregate score for the indicator and ultimately to the contributor.

◇ S2: **Building a Reference.** The reference dataset is used to evaluate QoI for a sample of indicators submitted by a sample provider. To build the initial reference dataset, data that is collected through security operations (e.g., monitoring, profiling, analyses, etc.) is vetted for their validity and applicability to the domain, perhaps using often expensive by necessary manual vetting [17, 19]. This reference dataset is used for the purpose of initializing the system.

◇ S3: **Extrapolating.** Extrapolation allows each assessor to predict the label of an indicator using its feature set and classifier model. The classifier is trained using a supervised learning process extracted from the reference dataset.

2.3. QoI Assessment Process

An illustration of the QoI assessment is depicted in Figure 1. The QoI assessment is achieved through a supervised learning process over the reference dataset, rather than the direct matching of explicit labels of indicators in the dataset.

We assume a reference labeled (training) dataset that contains a comprehensive library of artifacts, such as malware samples, incident reports, and logs, and that has been collected through operational intelligence gathering procedures. To predict a label correctly, the build of our trained model encompasses multiple components, namely a feature selection procedure, a machine learning algorithm selection procedure (e.g., SVM, logistic regression, random forest, etc.) and the corresponding parameters (e.g., procedure for regularization and linearization in case of SVM and LR, respectively), and cross-validation procedures (e.g., fold size, validation, etc.).

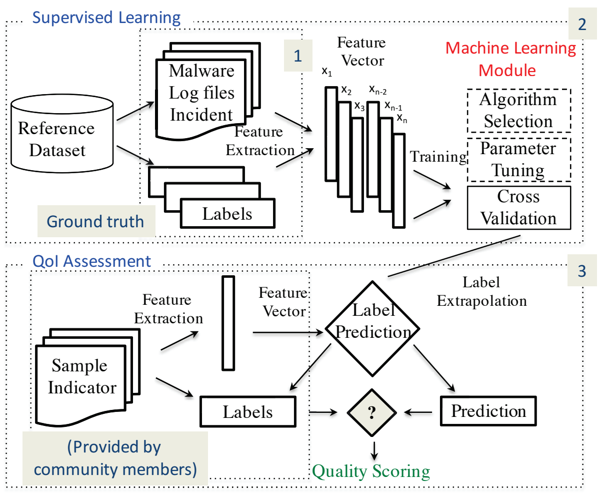

Fig. 1. The QoI assessment process, incorporating a reference model and a learning component for extrapolation.

The assessor uses the previously built model as a predictor and assigns a label to the indicator. The assessor decides the quality of the indicator by taking both the predicted and the self-provided labels into account (details are in the following section). The quality scoring procedure then aggregates the individual scores of the various indicators provided by each community member to assess their actual contribution.

2.4. Overall System Procedures

As outlined in sections 2.2 and 2.3, the QoI assessment process consists of multiple procedures for assessment initialization, learning, and extrapolation. In the following, we elaborate on some of the technical details of those procedures.

Reference Dataset and Learning. After identifying metrics for defining quality, we demonstrate the use of QoI for the assessment of the contribution level of participants. In order to initialize our system, a reference dataset is used to build a prediction model through supervised techniques of learning. Specifically, the collection of a reference dataset involves submission of sample artifacts from multiple sources, such as other community members, vendors, or own research [20].

We use various features for learning a model and their label prediction. For that, we use both static and dynamic analysis. For static analysis, we use artifacts such as file name, size, hashes, magic literals, compression artifacts, date, source, author, file type and portal executable (PE) header, among others. For dynamic analysis, we use counts that capture interactions between malware and the file system, user memory, registry, and network artifacts and features, as detailed in [20].

Modeling and Learning. In the following, we elaborate on a particular learning technique as a tool of QoI assessment.

To build our classifier, we obtain r samples for training from the reference dataset consisting of r_i training samples per class, with d features per sample. For each training sample y, we observe a label $\ell \in \Lambda$ and a sample vector $\vec{y}$, where $\vec{y}$

Procedure 1 Relevance of X_i (R)

```
1: Define weight values: $w_{r_1}, w_{r_2}, w_{r_3}, \dots w_{r_{|\Lambda|}} \in \mathbb{R}$
2: Define $w_R(.)$ to assign weights to labels: $w_R(l_i) = w_{r_i}$
3: Compute the relevance of $X_i$ as the average of the weighted sum:
   $R(X_i) = (\sum_{(\vec{x}_{ij}, l_{ij}) \in X_i} w_R(l_{ij})) / (\sum_{k=1}^{|\Lambda|} w_{r_k})$
```

is a vector of length m. We refer to the classes labels by their indices $i = 1, 2, \dots, \lambda$. We assume that samples labeled by i are distributed as $\mathcal{N}(\mu_i, \Sigma)$, the multivariate normal distribution with mean vector μ_i and standard deviation matrix Σ. We denote by $\mathcal{L}(x, \mu_i, \Sigma)$ the corresponding probability density function and by π_i the prior probability that an unknown sample comes from class labeled by i. Bayes' Theorem states that the probability an observed sample x comes from class i is proportional to the product of the class density and prior probability; namely $P(Z = i | X = x) \propto \mathcal{L}(x, \mu_i, \Sigma) \times \pi_i$, where $P(Z = i | X = x)$ is the posterior probability that sample x comes from class i. The classifier assigns the sample to the class with the largest posterior probability to minimize the misclassification error. This can be written as a rule: $\hat{z}(x) = \arg\min_i \{(x - \mu_i)^T \Sigma^{-1} (x - \mu_i) - 2\log(\pi_i)\}$ A sample is assigned nearest class with the distance being $||x - \mu_i||^2_\Sigma - 2\log(\pi_i)$, where $||x - \mu||^2 = (x - \mu)^T \Sigma^{-1} (x - \mu)$; square the Mahalanobis distance between x and μ.

Misclassification rate. A misclassification occurs when an indicator is assigned to an incorrect label, which done with $P(\epsilon)$, where: $P(\epsilon) = \sum_{j=1}^{\lambda} [P(\hat{Z} \neq j | Z = j) \times \pi_j]$.

Labeling and Quality Scoring. Denote by n the number of community members. Each (user) u_i provides a set of samples $X_i = \{(\vec{x}_{i1}, l_{i1}), (\vec{x}_{i2}, l_{i2}), (\vec{x}_{i3}, l_{i3}), \dots, (\vec{x}_{ik}, l_{ik})\}$ with feature vector $\vec{x}_{ij}$ and sample label $l_{ij} \in \Lambda$ for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, k$.

Correctness. The reference dataset is used as the benchmark for determining the correct label for an arbitrary sample. Each sample consists of a feature vector and an associated label. We first build a classifier by utilizing the reference dataset Y as the training set and forming a prediction on the label of $\vec{x}$ to obtain l'. Then, the assigned label of $\vec{x}$ is compared against the predicted label l' and a positive score is given if labels match. The correctness is computed as the average of the sum of scores for all samples in X_i.

Relevance. The algorithm for computing the relevance is shown in Procedure 1. In this procedure, the weight values $w_{r_1}, w_{r_2}, \dots w_{r_{|\Lambda|}}$ are chosen based on the context of use, and a mapping function $w_R(.)$ is defined to assign weights labels such that higher weight values are assigned to labels of greater interest to the community members. For each sample x, the corresponding label is evaluated using the mapping function $w_R(.)$ to obtain the weight value as the sample score. The relevance score of X_i, denoted by $R(X_i)$, is calculated as the average of the weighted sum of the scores.

Utility. Proc. 2 defines the utility. In this procedure we note that the utility of an indicator is determined by the sum of the utility weights of the samples. The weights $w_{t_1}, w_{t_2} \dots w_{t_d}$ and weight function $w_T(.)$ are defined by the application.

Procedure 2 Utility of X_i (U)

```
1: Define $t_1, t_2, \dots, t_d \in \mathbb{R}$.
2: Define weight values $w_{t_1}, w_{t_2} \dots w_{t_d}$, where each weight value corresponds to a utility type.
3: Define a weight function $w_T(.)$ s.t. $\ell \in \Lambda$ maps to a utility weight, i.e.
   $w_T(\ell) = w_{t_m}$ where $m = \{1, 2, \dots, d\}$.
4: for $x \in X_i$ do
5:     Compute a weight of $x = (\vec{x}, l')$, using $w_T(l') = w_{t_{l'}}$
6: end for
7: Compute the utility score of $X_i$ as the average of the sum of the sample
   weights: $U(X_i) = \frac{1}{k}\sum_{j=1}^{k} w_{t_j}$, where $t_j$ is the corresponding label
   type of sample $x_{ij} \in X_i$
```

Procedure 3 Uniqueness of X_i (N)

```
1: Consider the set $Z$ which is initially empty, i.e. $Z = \phi$
2: Build the set $Z$ by considering unique samples from the sets
   $X_1, X_2, ...X_n$
3: for $i = 1, 2, \dots n$ do
4:     for $j = 1, 2, \dots k$ do
5:         if $x_{ij} \notin Z$ then add $x_{ij}$ to $Z$.
6:         end if
7:     end for
8: end for
9: Compute $s_n(x_{ij}) = 1$ if $x_{ij} \in Z \backslash \{X_i\}$ and 0 otherwise.
10: Compute $N(X_i) = \frac{1}{k}\sum_{j=1}^{k} s_n(x_{ij})$
```

Uniqueness. Procedure 3 outlines the steps used for calculating the uniqueness of a set of indicators. In this procedure, we assume that samples can be uniquely identified (e.g. using hashes). In set notation, we can say that an element $x_{ij} \in X_i$ is unique if it is not an element of other sample sets, i.e. $x_{ij} \notin \bigcup X \backslash \{X_i\}$.

Quality of Indicator (QoI). QoI is a comprehensive measure and is calculated as the average of the weighted sum of the four components: correctness (C), relevance (R), utility (U) and uniqueness (N), as shown in procedure 4. The weights assigned for metrics are application-specific.

3. EVALUATION

Dataset. We use 11 malware families for our evaluation; five DDoS (Avzhan; 3458 samples, Darkness; 1878 samples, Ddoser; 502 samples, jkddos; 333 samples, N0ise; 431 samples), four targeted (ShadyRAT; 1287 samples, DNSCalc; 403 samples, Lurid; 399 samples, Getkys; 953 samples), and two mass market families (ZeroAccess; 568 samples, Zeus; 1975 samples). The dataset is collected between mid-2011 to mid-2013 [17]. Scans for labeling are done through VirusTotal around May 2013, static features are obtained from VirusTotal, and dynamic features are extracted using AMAL [20].

Results. We note that while there are more samples gathered for DDoS-type malware in comparison with others, the threat-intelligence community often gives more weight to identify malware or incidents that are less observable, which presents a level of sophistication. Thus, for our evaluation, we consider trojan and targeted malware more relevant than DDoS,

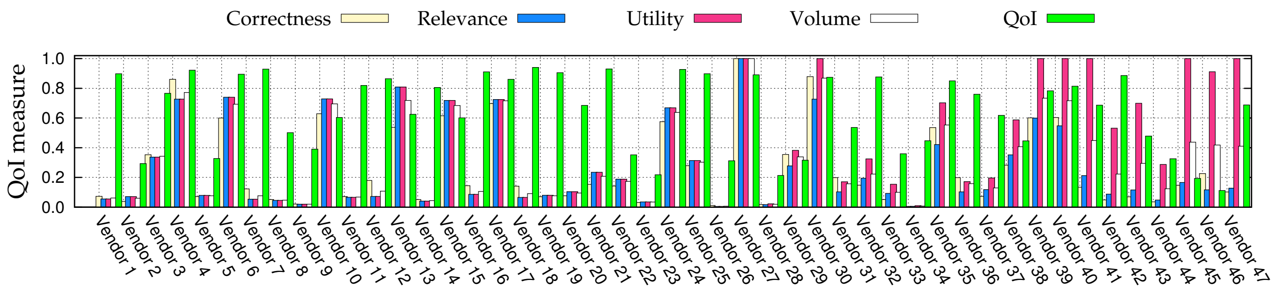

Fig. 2. Comparison between various quality-based metrics for AV indicator assessment

Procedure 4 QoI of X_i (QoI)

```
1: Define normalized weights, $w_C, w_R, w_U$, and $w_N$.
2: Calculate QoI as the weighted sum of the components: $QoI(X_i) = w_C C(X_i) + w_R R(X_i) + w_U U(X_i) + w_N N(X_i)$
```

from the point of view of community members consuming the shared information. The results are shown in Figure 2, and the following are observations on three of QoI metrics (since evaluating uniqueness is trivial).

◇ ***Correctness.*** As shown in Figure 2, the majority of other vendors have a gap in the score between the volume-based and the correctness-based contribution measures, where the correctness-based measure is significantly lower. Reasons may include labeling samples under unknown names, mislabeling due to similarities between families, or assigning generic labels like "trojan", "virus", and "unclassified". Examining the correctness of AV indicators also leads to a more subtle discussion about their utility. Looking closer into the label generated by some vendors, we found out that some labels are too generic and only describe the behavior rather than the name of a known malware family type, e.g., Trojan.Win32.ServStart vs. Avzhan.

◇ ***Relevance.*** We give more weight to targeted malware and Trojans over DDoS samples: $w_{targeted} = 5$, $w_{troj} = 3$, $w_{ddos} = 1$, and "0" otherwise. Thus, we observe two behaviors. First, certain contributors with high volume-based scores have a very low score (close to "0") of relevance. In particular, with the two relevant and one less relevant family types identified, such community members have more unidentified malware samples (e.g., vendor 7, vendor 21, vendor 59, etc.). On the other hand, other contributors with small volume-based contribution, have a higher relevance score, due to very relevant family identified in their shared indicator (e.g., vendor 10, vendor 16, vendor 27, etc.).

◇ ***Utility.*** To evaluate the utility, we give weights for three classes of malware labels: complete labels (w_c) are based on industrially popular name, generic labels (w_g) are based on placeholders commonly used for labeling the family such as " generic", " worm", " trojan", " start" and " run", and incomplete labels, (w_i), including " suspicious", " malware", and " unclassified", which do not hold any meaning of a class. Similar to the strategy with relevance, we assign weights of $w_c = 5$, $w_g = 2$, and $w_i = 1$.

◇ ***Aggregated* QoI *score.*** We aggregate a single QoI score for each vendor based on the weighted sum of the various QoI metrics. As can be seen, vendors, such as 39 and 46, with low QoI scores are rated with higher scores in their volume-based contribution, potentially alluding to free-riding. On the other hand, vendor 1, vendor 8, and vendor 33, among others, which tended to have lower volume-based scores, tend to have higher QoI, indicating their quality of contribution.

4. RELATED WORK

Adar and Huberman [1] identified the problem in P2P systems and noticed a large fraction of users who do not share useful content in the file sharing networks. Feldman *et al.* [7] characterized the problem of free-riding in peer-to-peer systems and proposed potential remedies. Locher et al. [12] developed a free-riding client as a proof-of-concept and demonstrated how files can be downloaded in the BitTorrent network without providing any content to peers. Others [9] focused on free-riding, and quality of contribution, by analyzing their root-cause and impact on the overall P2P utility. Related to QoI in threat intelligent systems is malware attribution [22]. Bailey et al. [2] were one of the early folks to characterize malware in terms of system state changes (e.g. registry changes, files created) and investigated the problem of behavior-based clustering as a method for classifying and analyzing Internet malware. Related to malware labeling, Mohaisen and Alrawi [17, 20] quantified the inconsistencies in labeling against a reference dataset collected from thousands of samples of various types manually vetted by analysts [26, 15, 23].

5. CONCLUSION

In this paper, we have the first look at the notion of the quality of indicators (QoI). As empirically analyzed, identifying the levels of contribution cannot be simply expressed in the volume-based measure of contribution. By verifying our metrics on a real-world data of antivirus scans we unveil that contribution measured by volume is not always consistent with those quality measures, and that QoI as a notion is capable of capturing forms of contribution analogous to free-riding.

Acknowledgement. This work is supported in part by NSF grant CNS-1809000, NRF grant NRF-2016K1A1A2912757, and the AFRL contract ICA2016-08-PIA3-007E.

6. REFERENCES

[1] E. Adar and B. A. Huberman. Free riding on gnutella. *First monday*, 5(10), 2000.

[2] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of internet malware. In *Proceedings of RAID*, 2007.

[3] S. Barnum. Standardizing cyber threat intelligence information with the structured threat information expression (stix^{TM}). *MITRE Corporation*, 11, 2012.

[4] S. Brown, J. Gommers, and O. Serrano. From cyber security information sharing to threat management. In *Proceedings of ACM WISCS*, 2015.

[5] L. Dandurand and O. S. Serrano. Towards improved cyber security information sharing. In *Proceedings of IEEE CyCon*, 2013.

[6] M. Feldman and J. Chuang. Overcoming free-riding behavior in peer-to-peer systems. *ACM SIGecom Exchanges*, 5(4):41–50, 2005.

[7] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. In *Proceedings of ACM SIGCOMM workshop*, 2004.

[8] J. C. Haass, G.-J. Ahn, and F. Grimmelmann. Actra: A case study for threat information sharing. In *Proceedings of WISCS*, 2015.

[9] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on gnutella revisited: the bell tolls? *IEEE Distributed Systems Online*, 6(6), 2005.

[10] J. Hur. Improving security and efficiency in attribute-based data sharing. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 25(10):2271–2282, 2013.

[11] P. Kampanakis. Security automation and threat information-sharing options. *IEEE Security & Privacy*, 12(5):42–51, 2014.

[12] T. Locher, P. Moor, S. Schmid, and R. Wattenhofer. Free riding in bittorrent is cheap. In *Proceedings of ACM HotNets*, 2006.

[13] E. Luiijf and M. Klaver. On the sharing of cyber security information. In *Proceedings of Springer ICCIP*, 2015.

[14] R. A. Martin. Making security measurable and manageable. In *Proceedings of IEEE MILCOM*, 2008.

[15] H. Mekky, A. Mohaisen, and Z.-L. Zhang. Separation of benign and malicious network events for accurate malware family classification. In *Proceedings of IEEE CNS*, 2015.

[16] A. Mohaisen, O. Al-Ibrahim, C. A. Kamhoua, K. A. Kwiat, and L. Njilla. Rethinking information sharing for threat intelligence. In *Proceedings of the ACM/IEEE HotWeb*, 2017.

[17] A. Mohaisen and O. Alrawi. *Detection of Intrusions and Malware, and Vulnerability Assessment: 11th International Conference, DIMVA 2014*, chapter AV-Meter: An Evaluation of Antivirus Scans and Labels, pages 112–131.

[18] A. Mohaisen and O. Alrawi. Unveiling zeus: automated classification of malware samples. In *Proc. of ACM WWW*, 2013.

[19] A. Mohaisen and O. Alrawi. AMAL: high-fidelity, behavior-based automated malware analysis and classification. In *Proceedings of WISA*, 2014.

[20] A. Mohaisen, O. Alrawi, and M. Mohaisen. Amal: High-fidelity, behavior-based automated malware analysis and classification. *ACM Computers & Security*, 52:251–266, 2015.

[21] S. Qamar, Z. Anwar, M. A. Rahman, E. Al-Shaer, and B.-T. Chu. Data-driven analytics for cyber-threat intelligence and information sharing. *Elsevier Computers & Security*, 67:35–58, 2017.

[22] C. Rossow, C. J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos, and M. Van Steen. Prudent practices for designing malware experiments: Status quo and outlook. In *Proceedings of IEEE SP*, 2012.

[23] F. Shen, J. Del Vecchio, A. Mohaisen, S. Y. Ko, and L. Ziarek. Android malware detection using complex-flows. In *Proceedings of IEEE ICDCS*, 2017.

[24] C. Sillaber, C. Sauerwein, A. Mussmann, and R. Breu. Data quality challenges and future research directions in threat intelligence sharing practice. In *Proceedings of WISCS*, 2016.

[25] F. Skopik, G. Settanni, and R. Fiedler. A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing. *Elsevier Computers & Security*, 60:154–176, 2016.

[26] J. Spaulding, S. Upadhyaya, and A. Mohaisen. The landscape of domain name typosquatting: Techniques and countermeasures. In *Proceedings of IEEE ARES 2016*, 2016.

[27] H. Tanaka, K. Matsuura, and O. Sudoh. Vulnerability and information security investment: An empirical analysis of e-local government in japan. *Elsevier Journal of Accounting and Public Policy*, 24(1):37–59, 2005.

[28] D. K. Tosh, S. Sengupta, C. A. Kamhoua, K. A. Kwiat, and A. P. Martin. An evolutionary game-theoretic framework for cyber-threat information sharing. In *Proceedings of IEEE ICC*, 2015.

[29] D. K. Tosh, S. Sengupta, S. Mukhopadhyay, C. A. Kamhoua, and K. A. Kwiat. Game theoretic modeling to enforce security information sharing among firms. In *Proceedings of IEEE CSCloud*, 2015.

[30] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody. Misp: The design and implementation of a collaborative threat intelligence sharing platform. In *Proceedings of WISCS*, 2016.