# A Decision Making Model for Collaborative Malware Detection Networks

Carol J. Fung, Disney Y. Lam, and Raouf Boutaba

School of Computer Science, University of Waterloo, Canada
{j22fung, y7lam, rboutaba}@uwaterloo.ca

Technical Report CS-2013-01

**Abstract.** The increased sophistication and evasiveness of malware has brought tremendous challenges to vendors of antivirus systems. Various malware detection approaches have been proposed and deployed to detect and remove malware. However, it is challenging for a single security vendor to analyze all malware and to provide up-to-date protection, e.g., a signature database. In this paper, we investigate the effectiveness of collaboration amongst various antivirus systems and propose a distributed collaborative malware detection network (CMDN). We design a novel collaborative malware detection decision model, RevMatch, where collaborative malware detection decisions are made based on the scanning history with multiple antivirus systems. We evaluate our system on real-world malware data sets and show that collaborative malware detection techniques can improve detection accuracy significantly. Furthermore, RevMatch outperforms existing decision models in terms of detection quality, runtime efficiency, and robustness against insider attacks.

## 1 Introduction

Cyber Intrusions have become more sophisticated and evasive. Each year, billions of cyber attacks are reported [24] and they cost hundreds of billions in losses [1]. Cyber attacks are typically accomplished with the assistance of *malware* (a.k.a. malicious code). Malware is a piece of software which is used to gather confidential information, exploit computing resources, or cause damage without users' consent. Typical examples of malware include worms, viruses, Trojan horses, spyware, and rootkits. Malware can spread through various routes, e.g., email attachments, Internet downloads, worms, or removable media.

Millions of new malware instances appear every year [12], and it has been growing at an exponential rate. Malware is used to not only harvest private information from compromised hosts, but also to organize such compromised hosts to form Botnets [3]. Many million-node Botnets have been discovered in the past few years, such as BredoLab [4] and Conficker [7]. Bots can be used to attack other hosts, such as Distributed-Denial-of-Services (DDoS) attacks. A recent DDoS attack in March 2013 targeting the largest spam filtering system, Spamhaus, is considered the largest DDoS attack in history [5]. The massive attacks generated traffic of 300Gbps and slowed the Internet down all around the world for one week.

To protect computers against malware, antivirus systems (AVs) are used to detect, block, and remove malware from hosts. Two typical metrics are used to measure the quality of an AV: the *true positive rate* (TP) and the *false positive rate* (FP). The former means an AV raises an alarm when there is a real threat; while the latter means an AV raises a false alarm for benign software. The goal of an AV is to maximize the TP rate while minimizing the FP rate. The most common technique to detect malware is *signature-based* detection, which involves searching for known malicious patterns within suspicious files. Signature-based detection performs fast and usually has a low FP rate. However, it may not be able to detect new threats, e.g., zero-day attacks. To mitigate such limitation, *heuristics-based* detection [17, 18] or *reputation-based* detection [2] is employed to improve malware detection efficiency. The heuristic approach analyzes malware and seeks similar patterns with known malicious code. The reputation-based approach evaluates the reputation of each file based on several attributes, such as file publisher, popularity, age, and reputation of host machines [10]. Both approaches are considered as a promising direction to detect new threats; however, heuristic matching without enough evidences of maliciousness can cause a high false positive rate.

Although the primary goal of an AV is to detect and remove malware, it is also important that malware detection system is able to correctly classify benign files. AVs with low TP rates may not effectively protect hosts from malware, while the consequences of false positives can be disastrous. For example, a security vendor released a flawed signature database update in 2010 which caused to remove a critical system file from Windows XP machines. The affected machines were no longer able to boot up [6]. TrendMicro spent $8 million reimbursing customers for reparation expenses [8].

Security vendors may not exchange information, e.g., malware samples reported from their customers, with other vendors because of privacy issues and competition. Providing prompt signature update against the latest threats is important to dominate a market. Isolated AVs cannot obtain malware samples of zero-day threats to be analyzed and may fail to protect their customers. However, from the customers' perspectives, if diverse security vendors collaborate with each other, by means of providing *feedback* regarding the legacy of suspicious files, they may achieve even better malware detection accuracy.

In this work, we investigate the effectiveness of AV collaboration and propose a fully-distributed collaborative malware detection network (CMDN) for AVs to exchange expertise, e.g., AVs send suspicious files or their hash values to other AVs for scanning and decide whether to raise an alarm or not based on feedback from other AVs. This paper focuses on the collaborative decision component design, with which our goal is to make accurate collaborative malware detection with acceptable runtime efficiency. We propose a new collaborative detection model named *RevMatch*, where the final malware decision is made based on looking up history with the same feedback combination. Our evaluation results, based on real-world malware, demonstrate that our algorithm effectively improves the detection accuracy compared to other decision algorithms in the literature, while it also performs well in runtime efficiency and other desired features. Although our framework is designed for AV collaborations, it can be also used for collaboration between intrusion detection systems.

The contribution of this paper can be summarized as follow: i) we propose a framework design for CMDN, where AVs help each other to improve malware detection efficiency, ii) we propose a novel collaborative decision algorithm named RevMatch and compare it with other existing approaches based on real-world malware samples. The results reveal the limitation of the current method of using AVs and the importance of AV collaboration, and iii) our collected evaluation data can be used as a benchmark by other researchers in the collaborative malware detection domain.

This paper is organized as follows: Section 2 discusses some existing collaborative malware detection systems and collaborative malware/intrusion detection decision methods. Section 3 discusses CMDN architecture design. The detailed design of collaborative decision model is described in Section 4. We present the evaluation results in Section 5 and further discuss the results in Section 6. Finally, we conclude this paper in Section 7.

## 2 Related Work

### 2.1 Collaborative malware detection

Using a collaborative approach for malware detection was previously discussed in the literature. Oberheide et.al. proposed CloudAV, a system [22] where end hosts send suspicious files to a central cloud-based anti-virus service for scanning malware with a number of different AVs. A threshold approach is used to aggregate feedback from multiple AVs. An implementation of CloudAV is described in [20]. RAVE [23] is another centralized collaborative malware scanning system where emails are sent to several "replicas" for malware scanning. A replica consists of a *payload*, which is running on one version of an AV for malware scanning, and a *wormhole*, which is used for collecting scanning results from a payload and commuting between different replicas for decision making. A simple voting based mechanism is employed to make final decisions.

Peer-to-peer communication overlay is also used for collaborative malware detection or general intrusion detection [19, 9, 14]. Decentralized network architectures allow participating nodes to share workload with others and thus avoid bottlenecks and single points of failure which are common weaknesses of centralized systems.

### 2.2 Decision models for collaborative malware detection

Several different models of collaborative decision for malware/intrusion detection have been proposed in the literature. We list a few that can be easily adapted to CMDN.

**Static Threshold** The static threshold (ST) model[22] raises an alarm if the total number of malware diagnosis in the result set is higher than a defined threshold. This model is straight forward and easy to implement. The tunable threshold can be used to decide the sensitivity in intrusion detection. However, the ST model considers the quality of all AVs equally, making the system vulnerable to attacks by colluded malicious insiders.

**Weighted Average** The weighted average (WA) model [21, 15] takes the weighted average of all feedback from AVs. If the weighted average is larger than the threshold, then the system raises an alarm. The weight of each AV can be the trust value or quality score of the AV. The impact from high-quality AVs is larger than from low-quality AVs. The Weighted Average model also provides a tunable threshold for the sensitivity of detection.

**Decision Tree** The decision tree (DT) model [11] uses a machine-learning approach to produce a decision tree, in order to maximize decision accuracy. The decision tree approach can provide a fast, accurate, and easy-to-implement solution to the collaborative malware detection problem. The training data with labeled samples is used to generate a binary tree and decisions are made based upon the tree. However, the decision tree approach does not work well with partial feedback, i.e., when not all participants give feedback. It is also not flexible (no easy way to tune the sensitivity of detection) since decision trees are usually precomputed.

**Bayesian Decision** The Bayesian decision (BD) model [16] is another approach for feedback aggregation in intrusion detection (or malware detection). In this approach, the conditional probability of malware/goodware given a set of feedback is computed using Bayes' theorem and the decision with the least risk cost is always chosen. The BD model is based on the assumption that feedbacks from collaborators are independent, which is usually not the case.

## 3 Collaboration Framework

In this section, we propose CMDN, a framework for AVs to perform collaborative malware detection. We then present the architecture design of CMDN and describe its components.

The topology of CMDN is shown in Fig. 1, where computers with malware detection capabilities are logically connected forming a peer-to-peer network. Each node maintains a list of collaborators to communicate with. We call the list of collaborators the *acquaintance list*. There are two different types of participating nodes in CMDNs: pure service nodes and trader nodes. *Pure service nodes* (e.g., cloud 1 in Fig. 1) only provide malware scanning services for others and do not request service from others. A pure service node may be an online malware scanning service provided by some security vendor or a system similar to CloudAV [22]. *Trader nodes* (e.g., AV5 in Figure 1), on the other hand, request services from other nodes and can also provide services in exchange if needed. Trading nodes allow participants to help each other by exchanging malware scanning services with each other.

The CMDN described above requires participating nodes to have malware scanning capabilities. When a node in the CMDN has a suspicious file detected by a heuristic or anomaly detector, but cannot make a confident decision about whether the file is malware (e.g., no matching malware signature is found), it may send the file or its digest to its acquaintances for scanning. When an acquaintance receives a malware scanning request, it either searches the past records for scanning results with the same digest, or analyzes the file and replies with a searching results or analysis result to the requester. Upon receiving feedback from its acquaintances, the requester decides whether to raise a malware alarm or not based on the aggregated feedbacks from its acquaintances (Section 4).

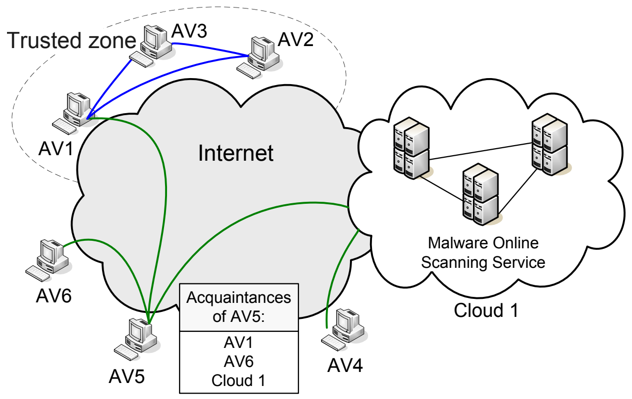


**Fig. 1.** Topology Design of Collaborative Malware Detection Network

## 3.1 Communication Protocol and Privacy Issue

To reduce the communication overhead in CMDNs, nodes may send the digest (fingerprint) of suspicious files first. If collaborator AVs find the digest in their blacklist/whitelist, then they return the corresponding result. Otherwise, they can request the sender to forward the original file.

When a host sends a file to its collaborators for scanning, the file receiver may hold the record and turn it against the sender. To reduce this privacy concern, original files are only sent to trusted peers for scanning in our CMDN design. To avoid man-in-the-middle attacks, all communication among connected nodes in a CMDN are encrypted to prevent eavesdropping.

The system also uses “test files” to evaluate the quality of collaborators and manage trust in the CMDN. The real scanning files and test files are sent randomly and it should be difficult for recipients to distinguish test files from real files.

## 3.2 Adversaries and Free-riders

Malicious insiders can be another issue in a CMDN since adversaries may disguise as an active CMDN participant and attack the CMDN. For example, adversaries may be sending false scanning results to other nodes or sending excessive scanning requests to others to overload the system. CMDNs can handle these problems by means of admission control and trust management. Trust management evaluates the expertise level and the honesty of nodes. Admission control restricts the amount of requests from participating nodes.

Free-riding is another potential problem in CMDN since it discourages nodes from contributing to the network. An incentive-compatible resource management encourages active contributors and discourages free-riding. Nodes who do not contribute to a CMDN shall be refrained from receiving assistance of other nodes in the network.

## 3.3 Architecture Design

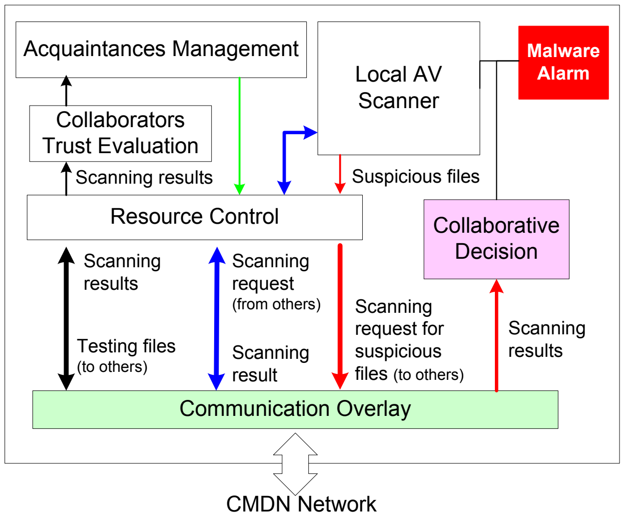


**Fig. 2.** Architecture Desgin of a Trader Node in CMDN

The architecture design of CMDN is illustrated in Fig. 2. Each node is composed of six components used for collaboration activities, namely, AV scanner, collaborative decision, communication overlay, resource control, trust evaluation, and acquaintance management.

The *Communication Overlay* is the component which handles all the communications between the host node and other peers in the network. The messages passing through the communication overlay include: test files from the host node to its acquaintances; malware consultation requests from the host node to its acquaintances; feedback from acquaintances; malware consultation requests from acquaintances; and feedback to acquaintances.

The *Collaborators Trust Evaluation* component allows AVs in the CMDN to evaluate the quality and trustworthiness of others. The host node can use test files to gain experience quickly. Indeed, the verified consultation results can also be used as experience.

The *Acquaintance Management* component decides who to collaborate with and manages different privileges for nodes with different trust levels. For example, nodes can send original files to trusted collaborators for scanning.

The *Resource Control* component is used to decide how much a host allocates resources to respond to the consultation requests from each of its acquaintances. An incentive-compatible resource management model can assist a node with an AV service to allocate resources to acquaintances in a fair manner. A node which abusively uses the resource of others will be penalized by being removed from the acquaintance lists of other nodes.

The *Collaborative Decision* component has a direct impact on the accuracy of the collaborative malware detection. After the host node sends out consultation requests to its acquaintances, the collected scanning results are used to decide whether the host should raise an alarm or not. Both false positive and false negative decisions bring costs to the host node. In the next section, we propose a decision model which can effectively improve collaborative detection accuracy.

## 4 Collaborative Decision Model

In this section, we present a collaborative decision algorithm named *RevMatch*, which can efficiently make collaborative malware detection decisions based on the feedback from acquaintances. In this model, each node in the CMDN keeps *labeled records* of its past experience with its acquaintances. Each labeled record contains the ground truth of a file (malware or goodware), a feedback set which contains the scanning results from the acquaintances, and the digest of the file (see Fig. 3). Labeled records can be obtained by sending test files to acquaintances for scanning. Past results from real file scanning requests can also be labeled once the ground truth of the file is revealed. In this section, we first formulate the collaborative decision problem an then propose our solution.

### 4.1 Problem Statement and RevMatch Model

We formulate the decision problem we are solving in this paper as follows:

*Given labeled records consisting of the feedback of $n$ AVs on $m$ files whose ground truth are known (malware or goodware), we decide whether a suspicious file is malware based on the feedback set $\mathbf{y}$ from a subset of $n$ AVs.*

We propose a decision model, RevMatch, which looks into past records for decisions. We formulate the decision problem as follows: suppose a scenario where $AV_i$ sends a suspicious file to the AVs in its acquaintance list $\mathcal{N}_i$ for scanning. Let variable $\mathbf{Y}_i := [Y_j]_{j\in\mathcal{N}_i}$ be the feedback vector that contains the scanning results from its acquaintances. Note that $Y_j = 1$ indicates the scanned file is a malware, and $Y_j = 0$ suggests a goodware[1]. Suppose $AV_i$ receives a feedback set $\mathbf{y} = \{\mathbf{y}_1, ..., \mathbf{y}_{|\mathcal{N}_i|}\}$ from its acquaintances, where $\mathbf{y}_j \in \{0, 1\}$, $j = 1, 2, \cdots, |\mathcal{N}_i|$. The observation $\mathbf{y}_j = 1$ indicates that the $j$-th acquaintance flags the file as malware, whereas $\mathbf{y}_j = 0$ as goodware. The problem is to decide whether $AV_i$ should decide whether the suspicious file is malware or not, based on the feedback $\mathbf{y}$. Table 1 summarizes the notations we use in this section for readers' convenience.

**Table 1.** Summary of Notations

| Symbol | Meaning |
|---|---|
| $n$ | Total number of AVs in the network. |
| $AV_i$ | Antivirus $i$. |
| $\mathcal{N}_i$ | Set of acquaintances of $AV_i$. |
| $\mathbf{M}, \mathbf{G}$ | Total number of malware and goodware in the labeled records database. |
| $\mathbf{P}_M, \mathbf{P}_G$ | Prior probability of malware and goodware in the real world. |
| $m$ | Total number of samples used for evaluation. $m = \mathbf{M} + \mathbf{G}$. |
| $Q_i$ | Quality score of $AV_i$. |
| $\mathbf{y}_k$ | Scanning results (feedback) from acquaintance $AV_k$. |
| $C_{fp}, C_{fn}$ | Cost of false positive and false negative decisions. |
| $\tau_s$ | The threshold for the static threshold method. |
| $\tau_w$ | The threshold for the weighted average method. |
| $\tau_c$ | Observation threshold for the RevMatch method. |
| $M(\mathbf{y}), G(\mathbf{y})$ | The number of malware and goodware in records with matching feedback set $\mathbf{y}$ |

We model the above decision problem as a utility optimization problem. Let random variable $X \in \{0, 1\}$ denote the outcomes of "goodware" and "malware". Let $\mathbf{P}_M(\mathbf{y})$ denote the probability of being "malware" given the feedbacks from all acquaintance AVs. $\mathbf{P}_M(\mathbf{y})$ can be written as $\mathbf{P}_M(\mathbf{y}) = \mathbb{P}[X = 1|\mathbf{Y} = \mathbf{y}]$. Let $C_{fp}$ and $C_{fn}$ denote the average cost of a FP decision and a FN decision. We assume that there is no cost when a correct decision is made. We define a decision function $\delta(\mathbf{y}) \in \{0, 1\}$, where $\delta = 1$ means raising a malware alarm and $\delta = 0$ means no alarm. The risk of decision $R(\delta)$ can be written as:

$$R(\delta) = C_{fn}\mathbf{P}_M(\mathbf{y})(1-\delta) + C_{fp}(1-\mathbf{P}_M(\mathbf{y}))\delta = (C_{fp} - (C_{fp}+C_{fn})\mathbf{P}_M(\mathbf{y}))\delta + C_{fn}\mathbf{P}_M(\mathbf{y})$$

To minimize the risk $R(\delta)$, we need to minimize $(C_{fp} - (C_{fp} + C_{fn})\mathbf{P}_M(\mathbf{y}))\delta$. Therefore, the AV raises malware alarm (i.e., $\delta = 1$) if

$$\mathbf{P}_M(\mathbf{y}) \geq \frac{C_{fp}}{C_{fp} + C_{fn}}. \tag{1}$$

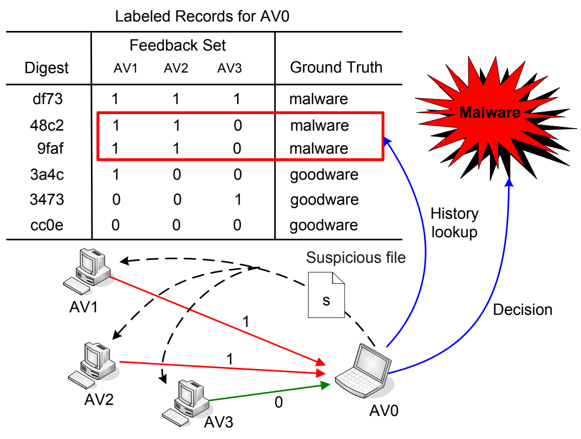
To make the optimal decision, the key step is estimating $\mathbf{P}_M(\mathbf{y})$. Our proposed solution is to search in the labeled records for records which have the same feedback set as $\mathbf{y}$. Let $M(\mathbf{y})$ and $G(\mathbf{y})$ denote the number of malware and goodware

[1] For the convenience of presentation, we drop the subscript $i$ in the notations appearing later in this paper.

in the labeled records with matching feedback set $\mathbf{y}$. If $M(\mathbf{y}) + G(\mathbf{y}) \geq \tau_c > 0$, then $\mathbf{P}_M(\mathbf{y})$ can be estimated using

$$\begin{aligned}
\mathbf{P}_M(\mathbf{y}) = \mathbb{P}[X = 1|\mathbf{Y} = \mathbf{y}] &= \frac{\mathbb{P}[\mathbf{Y} = \mathbf{y}|X = 1]\mathbb{P}[X = 1]}{\mathbb{P}[\mathbf{Y} = \mathbf{y}]} \\
&= \frac{\mathbb{P}[\mathbf{Y} = \mathbf{y}|X = 1]\mathbb{P}[X = 1]}{\mathbb{P}[\mathbf{Y} = \mathbf{y}|X = 1]\mathbb{P}[X = 1] + \mathbb{P}[\mathbf{Y} = \mathbf{y}|X = 0]\mathbb{P}[X = 0]} \\
&= \frac{\mathbb{P}[\mathbf{Y} = \mathbf{y}|X = 1]\mathbf{P}_M}{\mathbb{P}[\mathbf{Y} = \mathbf{y}|X = 1]\mathbf{P}_M + \mathbb{P}[\mathbf{Y} = \mathbf{y}|X = 0]\mathbf{P}_G} \\
&= \frac{1}{1 + \frac{\mathbb{P}[\mathbf{Y}=\mathbf{y}|X=0]\mathbf{P}_G}{\mathbb{P}[\mathbf{Y}=\mathbf{y}|X=1]\mathbf{P}_M}} \\
&\simeq \frac{1}{1 + \frac{G(\mathbf{y})\mathbf{M}\mathbf{P}_G}{M(\mathbf{y})\mathbf{G}\mathbf{P}_M}} \qquad (2)
\end{aligned}$$

where $P[\mathbf{Y} = \mathbf{y}|X = 1]$ is the probability that a feedback set $\mathbf{y}$ is received when the file is malware; $P[\mathbf{Y} = \mathbf{y}|X = 0]$ is the probability that diagnosis $\mathbf{y}$ is received when the file is goodware. $\mathbf{M}$ is the prior probability of malware; $\mathbf{G}$ is the prior probability of goodware. $\mathbf{P}_M, \mathbf{P}_G$ are the numbers of malware and goodware samples in the labeled records database.



Labeled Records for AV0

| Digest | Feedback Set AV1 | AV2 | AV3 | Ground Truth |
|---|---|---|---|---|
| df73 | 1 | 1 | 1 | malware |
| 48c2 | 1 | 1 | 0 | malware |
| 9faf | 1 | 1 | 0 | malware |
| 3a4c | 1 | 0 | 0 | goodware |
| 3473 | 0 | 0 | 1 | goodware |
| cc0e | 0 | 0 | 0 | goodware |

**Fig. 3.** An Example of the RevMatch Decision Algorithm for CMDNs

An example of this decision algorithm is illustrated in Figure 3. When $AV_0$ receives a suspicious file $s$ and cannot make a confident decision, it sends the file to its acquaintances $AV_1, AV_2, AV_3$ for scanning. The feedback set returned is $\{1, 1, 0\}$. $AV_0$ searches its labeled records database and finds two matches. Both matches are malware. Using the decision formula described in 2, $AV_0$ decides that file $s$ is a malware.

### 4.2 Feedback Relaxation

The previous results are based on the condition that $M(\mathbf{y}) + G(\mathbf{y}) \geq \tau_c$, where $\tau_c > 0$ is a system parameter to specify the minimum number of matches in order to reach some "confidence" in decision making using Eq. (2). In this subsection, we discuss how to deal with the case of $M(\mathbf{y}) + G(\mathbf{y}) < \tau_c$.

$M(\mathbf{y}) + G(\mathbf{y}) < \tau_c$ indicates there are not enough matches and thus no confident decision can be made. The RevMatch model handles this problem using *feedback relaxation*. That is, it ignores partial feedbacks from some acquaintances, intending to increase the number of matches by partial matching. The RevMatch model chooses to ignore the feedback from the least competent AV, since removing incompetent nodes can effectively increase the matching cases number while keeping valuable feedback from high quality AVs. The competence level of an AV can be its trust value or quality score.

Alg. 1 describes the process of removing incompetent AVs from the feedback set one by one until the number of matching samples exceeds the threshold $\tau_c$. Then, a decision is made based on the remaining feedback set. Upon receiving a diagnosis set $\mathbf{y}$, it first checks if the number of matching cases in the records exceeds the threshold $\tau_c$. If it does, it makes a decision based on the collected matches. Otherwise, the least competent AV is removed from the feedback set in each round until the number of matching samples exceeds the threshold. After that, it returns the corresponding decision and the remaining feedback set.

**Algorithm 1** Relaxation($\mathbf{y}$, $l_a$)

```
1: //This algorithm removes feedback from the least competent AVs from the acquaintances list until the number of matches reaches the threshold $\tau_c$. It has two parameters, the feedback vector $\mathbf{y}$ and an ordered list of AVs $l_a$, which is sorted by the competence levels of AVs in ascending order.
2: $(\mathbf{M}(\mathbf{y}), \mathbf{G}(\mathbf{y})) \Leftarrow$ find matches for $\mathbf{y}$
3: if $\mathbf{M}(\mathbf{y}) + \mathbf{G}(\mathbf{y}) \geq \tau_c$ then
4:     $\delta \Leftarrow \max_{\delta \in \{0,1\}} R(\delta)$
5:     return $(\mathbf{y}, \delta)$
6: end if
7: //Feedback relaxation
8: for each $a$ in $l_a$ do
9:     $\mathbf{y} \Leftarrow \mathbf{y}$ removes feedback of AV $a$
10:    $(\mathbf{M}(\mathbf{y}), \mathbf{G}(\mathbf{y})) \Leftarrow$ find matches for $\mathbf{y}$
11:    if $\mathbf{M}(\mathbf{y}) + \mathbf{G}(\mathbf{y}) \geq \tau_c$ then
12:        $\delta \Leftarrow \max_{\delta \in \{0,1\}} R(\delta)$
13:        return $(\mathbf{y}, \delta)$
14:    end if
15: end for
```

## 5 Evaluation

In this section, we use real data to evaluate the performance of the RevMatch model and compare it with four other decision models, namely, ST, WA, DT, and BD (described in Section 2). The metrics we use for the evaluation include detection accuracy, running time efficiency, and robustness against insider attacks. We use *quality score*, which is the combination of FP and FN, to measure detection accuracy; Running time efficiency is the average running time for making a decision; Robustness is the level of resistance to malicious insider attacks. We evaluate the performance of RevMatch and draw comparisons amongst different collaborative decision algorithms.

### 5.1 Data sets

In order to evaluate the accuracies of the decision algorithms, we collected real-world malware and goodware samples. Our malware data sets were collected from Malware Analysis System (formerly CW-Sandbox)[2], Offensive Computing[3], and other anti-virus vendors. In terms of the collection time, our malware datasets are divided into two groups: old malware data set (S1) collected in 2008–2009 and new malware data set (S2) collected in 2011–2012. We also mixed the two datasets and selected 50,000 of them to form a hybrid malware dataset (S3).

In our evaluation, we also included goodware to measure false positive rates of the decision algorithms. We crawled the top 10,000 projects in SourceForge[4] and extracted PE (Portable Executable) binary files as goodware samples (S4). We also collected binary files (S5) manually as false positive samples, such as some driver files and computer games from reputable producers from various sources. We also selected a mixed combination of goodware samples to form a hybrid goodware data set (S6). Table 2 shows the size of each data set.

**Table 2.** Data sets

| Data set ID | Data set | Samples | Date collected | Malware alarm rate |
|---|---|---|---|---|
| S1 | Old malware | 58,730 | 2008–2009 | 84.8% |
| S2 | New malware | 29,413 | 2011–2012 | 59.5% |
| S3 | Hybrid malware | 50,000 | 2009–2012 | 69.7% |
| S4 | Goodware (SourceForge) | 56,023 | 2012 | 0.3% |
| S5 | Goodware (Manual) | 944 | 2012 | 7.9% |
| S6 | Hybrid Goodware | 5,000 | 2012 | 1.6% |

We used VirusTotal[5] to obtain scanning results from a variety of anti-virus tools. Using the VirusTotal API, we uploaded our entire malware and goodware

[2] `https://mwanalysis.org/`
[3] `http://www.offensivecomputing.net/`
[4] `http://sourceforge.net/`
[5] `https://www.virustotal.com`

data sets and acquired scanning logs of 40 different anti-virus tools. Figure 4 shows both the TP and FP of each anti-virus engine based on hybrid datasets S3 and S6. One caveat is that we do not intend to compare different AV engines' detection rates because VirusTotal is not designed for performance comparisons. VirusTotal's scanning results are based upon command line versions of AV engines which may not be armed with more sophisticated techniques, e.g., behavioural analysis. We replace the names of AVs with indexed labels (e.g., $AV_i$) and the full list of AVs used in our experiments can be found in alphabetic order in Table 5.

We collected the average percentage of AVs raising malware alarms to each dataset based on VirusTotal's scanning results. We notice a higher percentage of AVs raise malware alarms on older malware samples than newer ones (see Table 2). The cause of the difference might be that antivirus vendors have more time to analyze and create more accurate anti-virus signatures for old malware samples.

In our setting, we used VirusTotal's scanning results as domain knowledge or previous observation on binary files. Given the same amount of information about binary files, our goal is to determine which decision algorithm i) yields the best detection rate and ii) provides more resilience against manipulated information.

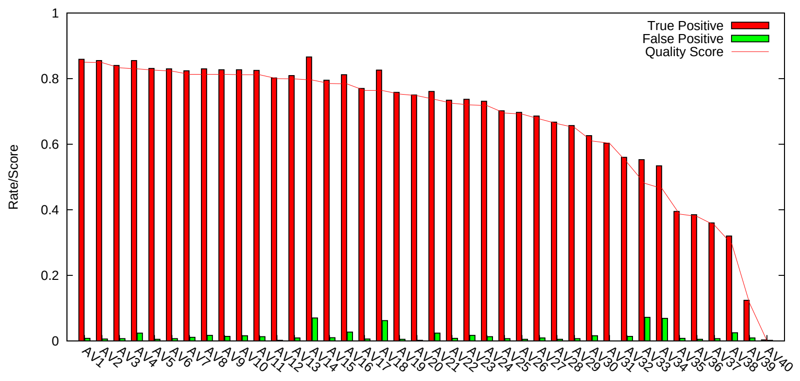


**Fig. 4.** True Positive Rate and False Positive Rate of AVs

## 5.2 Ranking of AVs

Both the WA model and RevMatch model require the ranking of AVs. In this section, we evaluate the TP, FP, and quality scores of AVs based on hybrid datasets S3 and S6. Moreover, the false negative rate (FN) is the probability that a malware is not detected and the true negative (TN) is the probability that goodware is correctly classified as goodware. High TP and low FP reflects high quality on malware detection. We define *quality score* of $AV_i$, denoted by

$Q_i$, using $Q_i = 1 - (C_{fn}FN_i + C_{fp}FP_i), \forall i \in \{1, 2, ..., n\}$, where $C_{fn}$ and $C_{fp}$ are the penalization factors on the false negative and false positive rates.

The FP, TP, and quality scores for all AVs are plotted in Fig. 4, where AVs are sorted by their quality scores. Complete data results can be found in [13](Appendix A). We can see that TP and FP from different AVs may vary greatly, and High quality AVs have both high TP and low FP. Results also show that all AVs are more effective in detecting old malware (S1) than new malware (S2).

## 5.3 Static Threshold

The static threshold (ST) model takes the total number of AVs which raises malware alerts. If the number is larger than a given threshold $\tau_s$, then it raises a malware alarm. i.e., if $\sum_{j \in \mathcal{N}_i} V_j \geq \tau_s$, where $V_j \in \{0, 1\}$ is the diagnosis result from $AV_j$, then rises a malware alarm.

We implemented the ST model and plot the evaluation results in Figure 5. We can see that FP decreases and FN increases when threshold $\tau_s$ raises. When $\tau_s$ is 0, ST reports all files to be malware; when $\tau_s$ is 40 (the total number of AVs), ST reports all files to be goodware. The quality score of ST reaches the highest when $\tau_s$ is 5. In the rest of this section, we set $\tau_s = 5$ unless we specify otherwise.

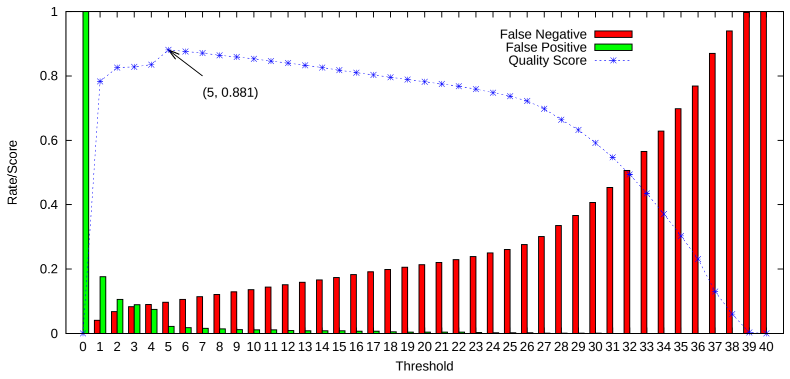


**Fig. 5.** TP, FP, and Quality Scores of Threshold-based Model with Different Thresholds (based on dataset S3, S6)

## 5.4 Weighted Average

The weighted average (WA) model takes the weighted average of the decisions from all AVs and asserts the suspicious file to be malware when the weighted average is higher than a threshold $\tau_w$. In our implementation, we use the quality scores computed in Section 5.2 as the weight of all AVs. i.e., WA only raises a malware alarm if $\frac{\sum_{j\in\mathcal{N}_i} Q_j V_j}{|\mathcal{N}_i|} \geq \tau_w$, where $V_j \in \{0,1\}$. As shown in Figure 6, WA yields optimal results when the threshold $\tau_w = 4/40$. Compared to ST, WA performs slightly better in malware detection quality. In the rest of the evaluation, we fix $\tau_w$ to $4/40$ unless we specify otherwise.

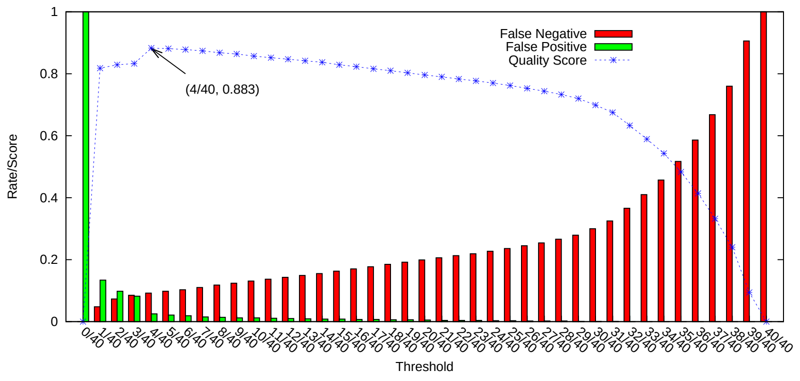


**Fig. 6.** TP, FP, and Quality Scores of Weighted Average Model with Different Thresholds (based on dataset S3, S6)

### 5.5 Decision Tree

The decision tree (DT) model uses machine learning to produce a tree-structured predictive tool to map feedback from different AVs to conclude that a suspicious file is a malware or not. We used Weka[6], a datamining software, as the machine learning tool to produce decision trees for evaluation. We chose algorithm $J48$ for decision tree generation based on dataset S3 and S6. We used 10-fold cross-validation to avoid overfitting. Figure 7 shows the partial outcome of the final decision tree. The entire decision tree includes 26 out of 40 AVs in the decision loop. Our results show that the DT model achieves a high TP 0.956. However, it also has a higher FP of 0.077, which leads to a moderate quality score of 0.879 (see Table 3). We speculate the reason behind this is that DT model focuses on reducing the overall number of false decisions, which does not necessary produce optimal *quality score* when there is large discrepancy in training data set sizes of malware and goodware.

### 5.6 Bayesian Decision

The Bayesian decision (BD) model uses Bayes' theorem to calculate the conditional probability $\mathbf{P}_M(\mathbf{y})$. A malware alarm is raised if $\mathbf{P}_M(\mathbf{y}) > \frac{C_{fp}}{C_{fp}+C_{fn}}$. However, the BD model is based on the assumption that all AVs are independent, which is not the case in reality. We also implemented the BD model and the detection accuracy is shown in Table 3.

[6] `http://www.cs.waikato.ac.nz/ml/weka/`

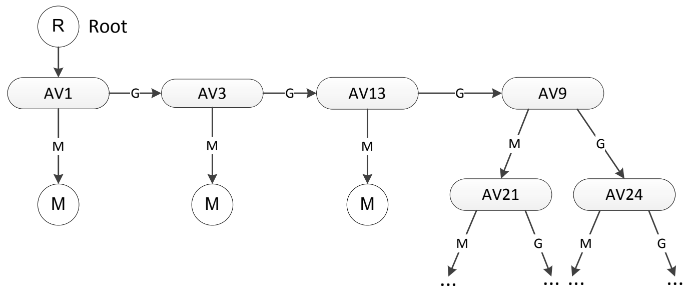


**Fig. 7.** The Optimal Decision Tree Generated by Weka *J*48 Algorithm (Top 5 levels)

## 5.7 RevMatch

The RevMatch model (Section 4) takes the feedback and does a history records look up for decision. We implemented RevMatch and evaluated it using 10-fold cross-validation based on datasets S3 and S6. In the first experiment, we fix parameters $C_{fp} = C_{fn} = 1$ and increase threshold $\tau_c$ from 1 to 5. As shown in Fig. 8, a higher $\tau_c$ leads to a slightly higher FN and lower quality score.

In the next experiment, we fix $\tau_c = 1$ and set different penalization weights on false negative rates $C_{fn}$. Figure 9 shows that a higher $C_{fn}$ leads to a higher FP and a lower FN. We speculate the reason is that RevMatch automatically trades FP for a lower FN, since the penalization of FN is higher.

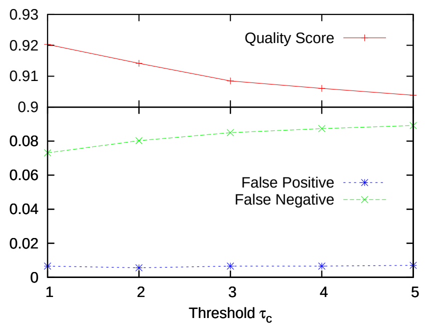


**Fig. 8.** The Impact from $\tau_c$ in RevMatch Model

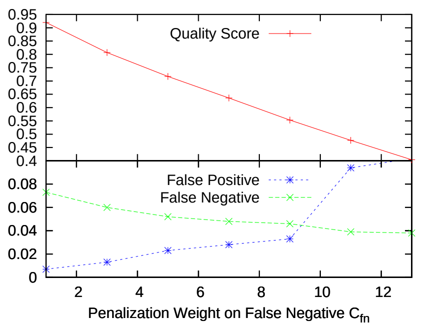


**Fig. 9.** The Impact from $C_{fn}$ in RevMatch Model

## 5.8 The Comparison between Different Decision Models

In this experiment, we compare the quality scores of five different decision models: ST, WA, DT, BD, and RevMatch. The results are based on dataset S3 and

S6. We used fixed thresholds 5 for ST and 4/40 for WA. We used 10-fold cross-validation for both DT and RevMatch models. We set parameter $\tau_c = 1$ and $C_{fp} = C_{fn} = 1$. The results are shown in Table 3. We can see that RevMatch outperforms all other models in terms of overall quality score. Also, all collaborative detection models have higher quality scores than any single AV.

Next, we increase $C_{fn}$ from 1 to 13 and plot the quality score of all decision models. We can see that RevMatch is superior to all others under all cases. BD performs the worst on higer $C_{fn}$. An interesting observation is that ST starts to perform better than WA when $C_{fn}$ is sufficiently large. We speculate the reason is that when it is costly to miss malware, then the system considers the opinion from all AVs rather than focusing on some high quality AVs. Note that in this experiment, ST and WA both re-select their optimal decision thresholds for each $C_{fn}$.

**Table 3.** Quality Scores Among Different Decision Models

| Method | True Positive TP | False Negative FN | False Positive FP | Quality Score $1 - C_{fp}FP - C_{fn}FN$ |
|---|---|---|---|---|
| Static Threshold | 0.903 | 0.097 | 0.022 | 0.881 |
| Weighted Threshold | 0.908 | 0.092 | 0.025 | 0.883 |
| Decision Tree | 0.956 | 0.044 | 0.077 | 0.879 |
| Bayesian Decision | 0.871 | 0.129 | 0.013 | 0.858 |
| RevMatch | 0.927 | 0.073 | 0.007 | 0.920 |
| Best Single AV | 0.859 | 0.141 | 0.008 | 0.851 |

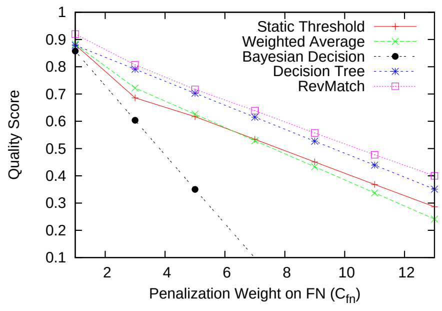


**Fig. 10.** Quality Scores of all Models With Different $C_{fn}$

## 5.9 Robustness against Insider Attacks

In an open CMDN, adversaries may join the network and serve as normal CMDN members in the beginning, and then suddenly turn around and send incorrect feedback. The tasks of quickly identifying and removing malfunctioning or malicious insiders are the responsibilities of trust management and acquaintance management components of the CMDN (Fig 2). In this subsection, we evaluate the impact of malicious insiders on the four decision models by intentionally injecting attacks into the data records.

In the first experiment, we start from the lowest ranking AV and replace its feedback by a malicious one, and gradually increase the number of malicious attackers by replacing feedback of other low quality AVs. We emulate three types of attacks, namely, the *alarmer attack*, the *dormant attack*, and the *random attack*. Attackers launching an alarmer attack always report malware whenever a scanning request is received; attackers launching a dormant attack always report goodware for all scanning requests; whereas in a random attack, nodes report random decisions (either malware or goodware). Figure 11 shows the impact of all these three different attacks on RevMatch model with different numbers of attackers. The alarmer attack has the highest impact and the dormant attack is the least effective. With the alarmer attack, the quality score drops down significantly when the number of attackers is higher than 5.

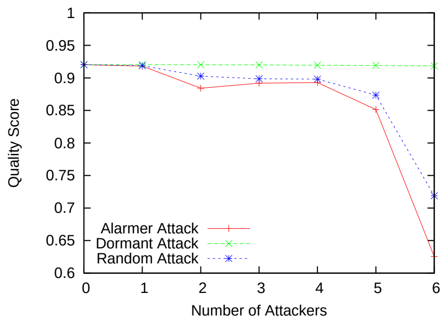


**Fig. 11.** RevMatch Model Under Three Different Attacks

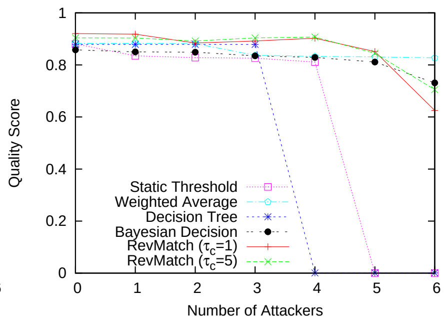


**Fig. 12.** The Quality Scores versus the Number of Attackers

In another experiment, we investigate the impact of alarmer attacks on different decision methods. Figure 12 shows that the decision tree was least durable to colluded alarmer attacks. Its quality score had no change with the first two attackers, but dropped quickly after the third attacker joined in. We investigated the reason and found that the first two AVs were not included in the decision tree while the third attacker AV was. The results also show that ST can endure at most 4 attackers since the decision threshold is 5. The RevMatch, BD, and WA models are relatively more robust to colluded alarmer attacks. We also notice that using a higher decision threshold $\tau_c$ on RevMatch increases the resistance against attackers while decreasing the detection quality when there is no insider attack.

## 6 Discussion

In the last section, we evaluated the performance of our proposed RevMatch model and compared it with four other collaborative decision models, namely, ST, WA, DT, and BD. The criteria we have used for evaluation are quality score and resistance to insider attacks. Quality score is a combination of FP and FN of the decisions, and the resistance to insider attacks is the maximum number of alarmer attackers it can endure before the quality score of the decision model drops significantly. In this section, we discuss other criteria that may be also important for choosing the right decision model for CMDN. They are: runtime efficiency, partial feedback adaption, and tuning flexibility.

### 6.1 Runtime Efficiency

Runtime Efficiency is an important criterion since it may not be acceptable for the system to take too long to make a decision. We evaluate the running time of all four decision models on a Ubuntu machine equipped with 2.13 GHz Intel Xeon and 3X4GB RAM. The ST, WA, BD, and DT models all take less than 1 milliseconds in processing the decision algorithm. RevMatch takes less than 15 milliseconds in average to make a decision.

### 6.2 Partial Feedback

In a CMDN, collaborators may not respond to scanning requests, especially when they are overloaded. Therefore, it is important for AVs to be able to make effective decisions based on the feedback from a subset of collaborators. ST may not work effectively with partial feedback since the fixed thresholds may be too high when the number of feedback participants is small. DT also does not work well with partial feedback, since it requires the inputs that can form a decision path in the tree. WA, BD, and RevMatch can work well with partial feedback.

### 6.3 Tuning Flexibility

Tunning flexibility allows the system administrator to tune the sensitivity of malware detection. For example, the system can become more or less sensitive to malware by changing a parameter. Both ST and WA can be tuned for the sensitivity of the system by setting their thresholds. DT, however, does not have a parameter that can be tuned for detection sensitivity. BD has tunning parameters $C_{fp}, C_{fn}$. RevMatch can be tuned using the penalization factors (i.e., $C_{fp}, C_{fn}$) for sensitivity, and $\tau_c$ for the robustness of the system.

### 6.4 Comparison

Table 4 provides a qualitative performance summary of the five collaborative decision models based on the metrics we selected. We can see that RevMatch is superior in terms of detection accuracy, flexibility, and adaptability to partial feedback. It also performs well in terms of runtime efficiency and resistance against insider attacks. Our results provide a reference for decision makers regarding which collaborative decision method to employ in their CMDNs.

**Table 4.** Performance Summary of Collaborative Decision Models

| Decision Model | Decision Quality | Runtime | Attacker Tolerance | Partial Feedback | Flexibility |
|---|---|---|---|---|---|
| Static Threshold | medium | very fast | 4 attackers | no | yes |
| Weighted Average | medium | very fast | 5+ attackers | yes | yes |
| Decision Tree | medium | very fast | 3 attackers | no | no |
| Bayesian Decision | low | very fast | 5+ attackers | yes | yes |
| RevMatch | high | fast | 5+ attackers | yes | yes |

## 7 Conclusion

In this paper, we presented a collaborative malware detection framework (CMDN) and focused on the design of its collaborative decision component. We proposed a decision model named RevMatch, which makes collaborative malware detection decision based on looking up the historical records with the same feedback set. We proposed several evaluation metrics and compared the RevMatch model with other decision models in the literature based on real data sets. Our evaluation results showed that RevMatch outperforms all others in terms of detection accuracy, flexibility, and tolerance of partial feedbacks, while achieving satisfactory running time efficiency and robustness to insider attacks. In general, collaborative malware detection techniques improve detection quality in comparison to single AVs. In our future work, we plan to further improve the robustness of the decision system by introducing more sophisticated insider attacks and devise corresponding defense mechanisms. We also intend to develop the trust management and acquaintance management components of the CMDN architecture.

## References

1. 2012 Norton Cybercrime Report. http://now-static.norton.com/now/en/pu/images/Promotions/2012/cybercrimeReport/2012_Norton_Cybercrime_Report_Master_FINAL_050912.pdf [Last accessed in April 5, 2013].
2. Antivirus vendors go beyond signature-based antivirus. http://searchsecurity.techtarget.com/magazineContent/Antivirus-vendors-go-beyond-signature-based-antivirus [Last accessed in April 5, 2013].
3. Bots and botnetsa growing threat. http://us.norton.com/botnet/promo [Last accessed in April 5, 2013].
4. Bredolab Bot Herder Gets 4 Years for 30 Million Infections. http://www.wired.com/threatlevel/2012/05/bredolab-botmaster-sentenced [Last accessed in April 5, 2013].
5. DDOS attack on Spamhaus: Biggest cyber-attack in history slows down internet across the world. http://www.mirror.co.uk/news/world-news/ddos-attack-spamhaus-biggest-cyber-attack-1788942 [Last accessed in April 5, 2013].
6. McAfee antivirus to reimburse consumers for bad update. http://news.techworld.com/security/3221657/mcafee-antivirus-to-reimburse-consumers-for-bad-update/ [Last accessed in April 5, 2013].
7. Protecting Against the Rampant Conficker Worm. http://www.pcworld.com /article/157876/protecting_against_the_rampant_conficker_worm.html [Last accessed in April 5, 2013].

8. Trend glitch costs 8 million. http://news.cnet.com/Trend-glitch-costs-8-million/2110-1002_3-5789129.html [Last accessed in April 5, 2013].
9. M. Cai, K. Hwang, Y.K. Kwok, S. Song, and Y. Chen. Collaborative internet worm containment. *IEEE Security & Privacy*, 3(3):25–33, 2005.
10. D Chau, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. Polonium: Tera-scale graph mining and inference for malware detection. In *Proccedings of SIAM International Conference on Data Mining (SDM) 2011*, 2011.
11. Y. Elovici, A. Shabtai, R. Moskovitch, G. Tahan, and C. Glezer. Applying machine learning techniques for detection of malicious code in network traffic. *KI 2007: Advances in Artificial Intelligence*, pages 44–50, 2007.
12. M. Fossi, D. Turner, E. Johnson, T. Mack, T. Adams, J. Blackbird, S. Entwisle, B. Graveland, D. McKinney, J. Mulcahy, et al. Symantec global internet security threat report. *XV, April*, 2010.
13. Carol J. Fung, Disney Y. Lam, and Raouf Boutaba. Revmatch: A decision model for collaborative malware detection. Technical Report CS-2013-01, Department of Computer Science, University of Waterloo, April 2013.
14. C.J. Fung and R. Boutaba. Design and management of collaborative intrusion detection networks. In *15th IFIP/IEEE Intl. Symp. on Integrated Network Management*, 2013.
15. C.J. 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.
16. C.J. Fung, Q. Zhu, R. Boutaba, and T. Barsar. Bayesian Decision Aggregation in Collaborative Intrusion Detection Networks. In *12th IEEE/IFIP Network Operations and Management Symposium (NOMS10)*, 2010.
17. Jiyong Jang, David Brumley, and Shobha Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 309–320. ACM, 2011.
18. Dmitriy Komashinskiy and Igor Kotenko. Malware detection by data mining techniques based on positionally dependent features. In *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*, pages 617–623. IEEE, 2010.
19. Mirco Marchetti, Michele Messori, and Michele Colajanni. Peer-to-peer architecture for collaborative intrusion and malware detection on a large scale. *Information Security*, pages 475–490, 2009.
20. Cristian Adrián Martínez, Gustavo Isaza Echeverri, and Andrés G Castillo Sanz. Malware detection based on cloud computing integrating intrusion ontology representation. In *Communications (LATINCOM), 2010 IEEE Latin-American Conference on*, pages 1–6. IEEE, 2010.
21. P. Miller and A. Inoue. Collaborative intrusion detection system. In *Fuzzy Information Processing Society, 2003. NAFIPS 2003. 22nd International Conference of the North American*, pages 519–524. IEEE, 2003.
22. J. Oberheide, E. Cooke, and F. Jahanian. Cloudav: N-version antivirus in the network cloud. In *Proc. of the 17th USENIX Security Symp.*, 2008.
23. Carlos Silva, Paulo Sousa, and Paulo Verissimo. Rave: Replicated antivirus engine. In *Dependable Systems and Networks Workshops (DSN-W), 2010 International Conference on*, pages 170–175. IEEE, 2010.
24. Paul Wood, Mathew Nisbet, Gerry Egan, Nicholas Johnston, Kevin Haley, Bhaskar Krishnappa, Tuan-Khanh Tran, Irfan Asrar, Orla Cox, Sean Hittel, et al. Symantec internet security threat report trends for 2011. *Volume XVII*, 2012.

## A Appendix A

**Table 5.** Antiviruses Used for Evaluation (presented in alphabetical order)

| | | | |
|---|---|---|---|
| AhnLab-V3 | Comodo | Jiangmin | Rising |
| AntiVir | DrWeb | K7AntiVirus | Sophos |
| Antiy-AVL | Emsisoft | Kaspersky | SUPERAntiSpyware |
| Avast | eSafe | McAfee | Symantec |
| AVG | eTrust-Vet | Microsoft | TheHacker |
| BitDefender | Fortinet | NOD32Norman | TrendMicro |
| ByteHero | F-Prot | nProtect | VBA32 |
| CAT-QuickHeal | F-Secure | Panda | VIPRE |
| ClamAV | GData | PCTools | ViRobot |
| Commtouch | Ikarus | Prevx | VirusBuster |

**Table 6.** Quality Ranking for Antiviruses (AV1-AV40 correspond to the AVs listed in Table 5 with assigned nick names)

| | Antivirus Name | Detection Rate (old malware S1) | Detection Rate (new malware S2) | True Positive (malware S3) | False Positive (goodware S6) | Quality Score $C_{fn} = C_{fp} = 1$ |
|---|---|---|---|---|---|---|
| 1 | AV1 | 0.951 | 0.800 | 0.859 | 0.008 | 0.851 |
| 2 | AV2 | 0.944 | 0.797 | 0.855 | 0.006 | 0.849 |
| 3 | AV3 | 0.925 | 0.787 | 0.840 | 0.007 | 0.833 |
| 4 | AV4 | 0.961 | 0.783 | 0.855 | 0.024 | 0.831 |
| 5 | AV5 | 0.939 | 0.759 | 0.831 | 0.005 | 0.826 |
| 6 | AV6 | 0.939 | 0.757 | 0.830 | 0.007 | 0.823 |
| 7 | AV7 | 0.940 | 0.747 | 0.824 | 0.011 | 0.813 |
| 8 | AV8 | 0.946 | 0.752 | 0.830 | 0.017 | 0.813 |
| 9 | AV9 | 0.952 | 0.742 | 0.827 | 0.014 | 0.813 |
| 10 | AV10 | 0.932 | 0.755 | 0.827 | 0.016 | 0.812 |
| 11 | AV11 | 0.936 | 0.752 | 0.825 | 0.013 | 0.812 |
| 12 | AV12 | 0.914 | 0.733 | 0.802 | 0.002 | 0.800 |
| 13 | AV13 | 0.931 | 0.726 | 0.809 | 0.009 | 0.799 |
| 14 | AV14 | 0.947 | 0.813 | 0.866 | 0.070 | 0.796 |
| 15 | AV15 | 0.863 | 0.753 | 0.795 | 0.010 | 0.785 |
| 16 | AV16 | 0.935 | 0.726 | 0.812 | 0.027 | 0.784 |
| 17 | AV17 | 0.931 | 0.654 | 0.770 | 0.006 | 0.764 |
| 18 | AV18 | 0.908 | 0.779 | 0.826 | 0.062 | 0.764 |
| 19 | AV19 | 0.911 | 0.648 | 0.758 | 0.005 | 0.753 |
| 20 | AV20 | 0.891 | 0.653 | 0.750 | 0.002 | 0.748 |
| 21 | AV21 | 0.890 | 0.679 | 0.761 | 0.024 | 0.737 |
| 22 | AV22 | 0.927 | 0.594 | 0.734 | 0.008 | 0.725 |
| 23 | AV23 | 0.938 | 0.607 | 0.737 | 0.017 | 0.720 |
| 24 | AV24 | 0.929 | 0.592 | 0.731 | 0.013 | 0.718 |
| 25 | AV25 | 0.903 | 0.562 | 0.702 | 0.007 | 0.695 |
| 26 | AV26 | 0.907 | 0.556 | 0.697 | 0.005 | 0.692 |
| 27 | AV27 | 0.897 | 0.544 | 0.686 | 0.009 | 0.677 |
| 28 | AV28 | 0.849 | 0.546 | 0.667 | 0.005 | 0.663 |
| 29 | AV29 | 0.882 | 0.513 | 0.657 | 0.007 | 0.651 |
| 30 | AV30 | 0.861 | 0.461 | 0.626 | 0.016 | 0.610 |
| 31 | AV31 | 0.755 | 0.494 | 0.603 | 0.000 | 0.603 |
| 32 | AV32 | 0.771 | 0.421 | 0.560 | 0.014 | 0.545 |
| 33 | AV33 | 0.814 | 0.377 | 0.553 | 0.072 | 0.481 |
| 34 | AV34 | 0.746 | 0.416 | 0.534 | 0.069 | 0.465 |
| 35 | AV35 | 0.525 | 0.330 | 0.395 | 0.008 | 0.387 |
| 36 | AV36 | 0.754 | 0.141 | 0.385 | 0.005 | 0.380 |
| 37 | AV37 | 0.474 | 0.283 | 0.360 | 0.007 | 0.353 |
| 38 | AV38 | 0.473 | 0.221 | 0.320 | 0.025 | 0.295 |
| 39 | AV39 | 0.204 | 0.062 | 0.124 | 0.009 | 0.116 |
| 40 | AV40 | 0.022 | 0.001 | 0.003 | 0.002 | 0.001 |