RevMatch: An Efficient and Robust Decision Model for Collaborative Malware Detection

Carol J. Fung[1], Disney Y. Lam[2], and Raouf Boutaba[2]
[1]Computer Science Department,
Virginia Commonwealth University, Virginia, USA,
cfung@vcu.edu
[2]David R. Cheriton School of Computer Science,
University of Waterloo, Ontario, Canada,
{y7lam,rboutaba}@uwaterloo.ca

***Abstract*—This work falls in the area of collaborative malware detection systems which rely on expertise and knowledge from multiple different antivirus software for malware detection. A critical component of such systems is the collaborative malware detection decision process. In this paper, we propose a novel decision model, RevMatch, where collaborative malware decisions are made based on labeled malware detection history from participating antiviruses. We evaluate our proposal using real-world malware data sets and demonstrate that collaborative malware detection techniques can improve the malware detection accuracy compared to using a single albeit the best antivirus. Moreover, we demonstrate how RevMatch outperforms all other existing collaborative decision models in terms of detection accuracy while being computationally efficient and robust against various malicious insider attacks.**

I. Introduction

Malware has become more sophisticated and evasive. Millions of new malware instances appear every year [10] 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 hosts to form Botnets. Many million-node Botnets have been discovered in the past few years, such as BredoLab [2] and Conficker [5]. Bots can be used to attack other hosts in Distributed-Denial-of-Services (DDoS) attacks. A recent well-known DDoS incident occurred in late 2012, where a series of DDoS attacks was launched against the American financial services sector, which led to damages of $30,000 each minute when their services were down [3]. A recent DDoS attack in March 2013 targeting the largest spam filtering system, Spamhaus, generated traffic of 300Gbps and slowed the Internet down globally 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 while minimizing the FP.

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. However, it may not be able to detect new threats, e.g., zero-day attacks. To mitigate such limitation, *behaviour-based* detection [11], [15], *heuristics-based* detection [14], [16] and *reputation-based* detection [1] are employed to improve malware detection efficiency. The behaviour-based approach analyzes behaviour log or graph of a suspicious file, such as a sequence of system calls, and matches them with known malware behaviour patterns. The heuristic approach analyzes malware and seeks similar patterns with known malware. 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 [8]. All three approaches are considered as a promising direction to detect new threats; however, heuristic matching without sufficient evidences of maliciousness can cause a high FP.

Although the primary goal of an AV is to detect and remove malware, it is equally important that malware detection system is able to correctly classify benign files. AVs with low TP may not effectively protect hosts from malware, while the consequences of false positives can be disastrous. For example, in 2010, a security vendor released a flawed signature database update which removed a critical system file from Windows XP machines, causing the affected machines to be unable to boot up afterward. [4]. In a similar instance, TrendMicro spent $8 million reimbursing customers for reparation expenses [6].

An AV vendor who may be effective in detecting one type of malware may be ineffective at detecting other types of malware. Additionally, an AV vendor may fail to obtain samples of zero-day malware in time for analysis and thus fail to protect their customers. However, if diverse AV vendors collaborate with each other, by providing *feedback* about suspicious files or activities, they may achieve better malware detection accuracy and in turn better satisfy their customers. A collaborative malware detection system (CMD) allows antivirus systems to use collective knowledge and expertise from each other for malware detection. Several similar systems have been proposed in the literature, including CloudAV [19], RAVE [20], and CIDN [12].

A key component for a CMD is the collaborative malware detection, where the decision of a suspicious file be-

ing benign or malicious is based on opinions from multiple AVs. Existing models adopted in CMDs include static threshold [19], weighted average [12], decision tree [9], and Bayesian model [13]. However, no work has been done to evaluate and compare their performances against real malware data.

In this work, we propose a novel collaborative decision model named *RevMatch*, where the final malware decision is made based on querying history with the same feedback combination. When such a match is not found or the number of matches is too small for a confident decision, then partial matching is sought instead. Our evaluation results, based on real-world malware data sets, demonstrate that our algorithm effectively improves detection accuracy compared to other decision algorithms in the literature, while it also performs well in term of runtime efficiency and is robust to malicious insiders.

This paper is organized as follows: Section II discusses some existing collaborative malware detection systems and collaborative malware/intrusion detection decision methods. The detailed design of the collaborative decision model is described in Section III. We present the evaluation results in Section IV and further discuss the results in Section V. Finally, we conclude this paper in Section VI.

II. Related Work

A. Collaborative malware detection

Using a collaborative approach for malware detection was previously discussed in the literature. Oberheide et.al. proposed CloudAV, a system [19] where end hosts send suspicious files to a central cloud-based antivirus service for scanning by a number of different AVs. A threshold approach is used to aggregate feedback from multiple AVs. An implementation of CloudAV is described in [18]. RAVE [20] is another centralized collaborative malware scanning system where emails are sent to several agents for malware scanning. A simple voting based mechanism is employed to make final decisions.

Peer-to-peer communication overlay is also used for collaborative malware detection [17], [7], [12]. Decentralized network architectures allow participants to share workload with others and thus avoid bottlenecks and single points of failure which are common weaknesses of centralized systems.

B. Decision models for collaborative malware detection

Several collaborative decision models for malware/intrusion detection have been proposed in the literature. Here we discuss a few that have been proposed for CMDs.

1) Static Threshold: The static threshold (ST) model [19] 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 malware detection. However, the ST model considers the quality of all AVs equally, making the system vulnerable to attacks by colluded malicious insiders.

2) Weighted Average: The weighted average (WA) model [12] 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 WA model also provides a tuneable threshold for the sensitivity of detection.

3) Decision Tree: The decision tree (DT) model [9] 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.

4) Bayesian Decision: The Bayesian decision (BD) model [13] 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.

III. Collaborative Decision Model

In a CMD, a collaborative malware detection decision model based on feedback is key to obtaining high detection accuracy. Efficiency (malware detection accuracy) is the primary goal of a decision model for CMD. Robustness is also important since adversaries have strong motivation to evade or compromise the system. However, robustness is not discussed in most machine-learning approaches. In this paper, we propose an efficient and robust collaborative decision model named *RevMatch*, which can make accurate collaborative malware detection decisions based on the feedback from neighbors, and which is also robust to malicious insider attacks. In this section, we first formulate the collaborative decision problem and then describe our solution.

A. Problem Statement and RevMatch Model

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

Given a labeled history 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 the AVs.

To solve this problem, we model the decision problem as follows. Suppose a scenario where a set of AVs $\mathcal{N}$ are consulting each other for malware assessment. AV_i ($i \in \mathcal{N}$) sends a suspicious file to other AVs in its neighbor list $\mathcal{N}_i$ for consultation. Let random variable $\mathbf{Y}_i := [Y_j]_{j \in \mathcal{N}_i}$ denote the feedback vector that contains the scanning results

from the neighbors. Note that $Y_j \in \{0,1\}$, and $Y_j = 1$ and $Y_j = 0$ indicate the suspicious file is a malware or goodware respectively[1]. Suppose AV_i sends a suspicious file to its neighbors for consultation and receives a feedback set $\mathbf{y} = \{\mathbf{y}_1, ..., \mathbf{y}_{|\mathcal{N}_i|}\}$ from its neighbors, where $\mathbf{y}_j \in \{0,1\}$ is the feedback from neighbor j. AV_i needs to decide whether the suspicious file is malware or not, based on the feedback $\mathbf{y}$.

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 y from all neighbor 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:

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

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 to estimate $\mathbf{P}_M(\mathbf{y})$. Our proposed solution (RevMatch) is to search in the labeled history 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 in the labeled records with matching feedback set $\mathbf{y}$. If the number of observed matching records in history is larger than a threshold, i.e., $M(\mathbf{y}) + G(\mathbf{y}) \geq \tau_c > 0$, then $\mathbf{P}_M(\mathbf{y})$ can be estimated using

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

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. $\mathbf{P}_M$ is the prior probability of malware; $\mathbf{P}_G$ is the prior probability of goodware. $\mathbf{M}, \mathbf{G}$ are the numbers of malware and goodware samples in the labeled history.

We use a simple example in Fig. 1 to illustrate a use case of this decision model. When AV_0 receives a suspicious file s and cannot make a confident decision, it sends the file to its neighbors 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. If $\tau_c = 2$, AV_0 decides that file s is malware using the decision formula described in Eq. (2).

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

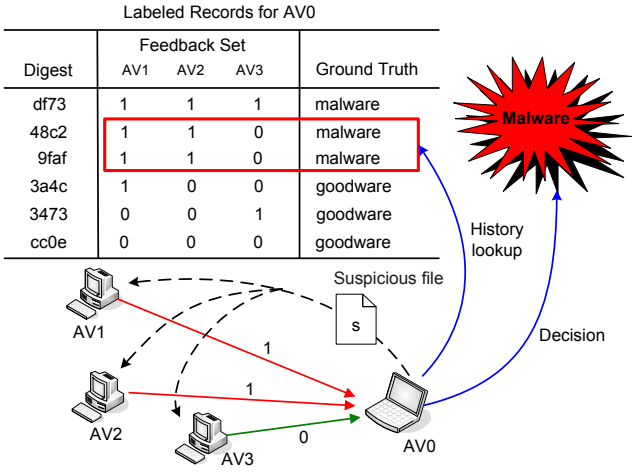

Fig. 1. An Example of the RevMatch Decision Algorithm for CMDs

B. 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 feedbacks from some neighbors, 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 τ_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 τ_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)

```
//This algorithm removes feedback from the least competent AVs from the neighbors list until the number of matches reaches the threshold τ_c. It has two parameters, the feedback vector y and an ordered list of AVs l_a, which is sorted by the competence levels of AVs in ascending order.
(M(y), G(y)) ⇐ find matches for y
if M(y) + G(y) ≥ τ_c then
    δ ⇐ max_{δ∈{0,1}} R(δ)
    return (y, δ)
end if
//Feedback relaxation
for each a in l_a do
    y ⇐ y removes feedback of AV a
    (M(y), G(y)) ⇐ find matches for y
    if M(y) + G(y) ≥ τ_c then
        δ ⇐ max_{δ∈{0,1}} R(δ)
        return (y, δ)
    end if
end for
```

Algorithm 2 Ground Truth Update($\mathbf{T}, \bar{F}, \bar{g}$)

```
//This algorithm updates the ground truth set T when a new ground truth ḡ with scanning feedback F̄ arrives.
j ⇐ search records in T with feedback F̄
if j ≥ 0 and t_j < currentTime() − Δt then
    a_j ⇐ αa_j + ḡ // update the number of malware
    b_j ⇐ αb_j + (1 − ḡ) // update the number of goodware
else if j is not found then
    F_new = F̄ // create a new entry F_new = F̄
    a_new ⇐ βP_M + ḡ
    b_new ⇐ βP_G + (1 − ḡ)
    T ⇐ T ∪ {F_new, a_new, b_new, currentTime()}
end if
```

C. Labeled History Update

The *labeled history* (ground truth set) is highly important since all decisions are based on the ground truth (GT) search for matches. AVs in CMD may collect labeled history by sending test files to neighbors and recording their feedbacks and GT. Real consultation files can also be used when their GT are revealed afterward.

The GT set $\mathbf{T}$ is a collection of feedback records labeled with their GT (malware or goodware) as shown in Fig. 1. To increase storage efficiency, a GT entry $\mathbf{T}_i$ can be represented with attributes $\{F_i, a_i, b_i, t_i\}$. F_i is the binary set representing the feedbacks from neighbors, a_i and b_i are the number of malware and goodware in history with feedback F_i. t_i is the timestamp of the last GT sample recorded with feedback F_i. The purpose of recording the timestamp is to prevent history poison flooding attacks, where a malicious insider (probably a malware producer) accumulates credibility quickly by releasing a large number of zero-day malware that other AVs may not be able to detect in the beginning, and then raises alarms on goodware to mislead others (see Section V-F).

The labeled history update process is described in Alg. 2. When a node has a new test file with GT $\bar{g} \in \{0, 1\}$, it sends the file to all collaborators for consultation and receives feedback $\bar{F}$. Suppose there exists an entry $F_j = \bar{F}$ in the labeled history and $t_j < currentTime() - \Delta t$, then update $a_j = \alpha a_j + \bar{g}$ and $b_j = \alpha b_j + (1 - \bar{g})$, and also reset t_j; otherwise if there is no entry with feedback $\bar{F}$, then create a new entry $\{F_{new}, a_{new}, b_{new}, t_{new}\}$. Δt is the minimum time gap that two adjacent updates have the same feedback. α is the discount factor on older data and β is the weight on priors. $\mathbf{P}_M$ and $\mathbf{P}_G$ are the priors for malware and goodware.

IV. 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 II). 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 (e.g., 1-FN-FP), 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 compare its performance with different collaborative decision algorithms.

TABLE I
Data sets

Dataset ID	Dataset description	Samples	Year	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%

A. 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 antivirus 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 I shows the size of each data set.

We used VirusTotal[5] to obtain scanning results from a vari-

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

ety of antivirus tools. Using the VirusTotal API, we uploaded our entire malware and goodware data sets and acquired scanning logs of 40 different antivirus tools. Fig. 2 shows both the TP and FP of each antivirus 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 this type of performance comparison. VirusTotal's scanning results are based upon command line versions of AV engines which may not be equipped 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 II.

TABLE II
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

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

B. Experiment Setting

We emulated a CMD composed of 40 AVs from different vendors. The data collected in Section IV-A are partially used for constructing labeled history for nodes in CMD. The left data are used for testing/evaluation. In the next subsections, we evaluate and compare the efficiency of several different collaborative decision models.

C. 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 of malware detection. We define *quality score* of AV_i, denoted by Q_i, using

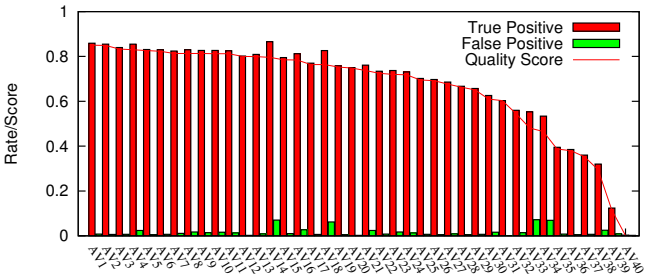

Fig. 2. True Positive Rate and False Positive Rate of AVs

$Q_i = 1-(C_{fn}FN_i + C_{fp}FP_i), \forall i \in \{1,2,...,n\}$, 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. 2, where AVs are sorted by their quality scores ($C_{fn} = C_{fp} = 1$). We can see that TP and FP from different AVs vary greatly, and high quality AVs have both high TP and low FP. The highest quality score a single AV can achieve is 0.851.

D. Static Threshold

The static threshold (ST) model takes the total number of AVs which raise malware alerts. If the number is larger than a given threshold τ_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 it raises a malware alarm.

We implemented the ST model and plot the evaluation results in Fig. 3. We can see that FP decreases and FN increases when threshold τ_s increases. When τ_s is 0, ST reports all files to be malware; when τ_s is 40 (the total number of AVs), ST reports all files to be goodware. The quality score of ST reaches the highest when τ_s is 5. In the rest of this section, we set $\tau_s = 5$ unless we specify otherwise.

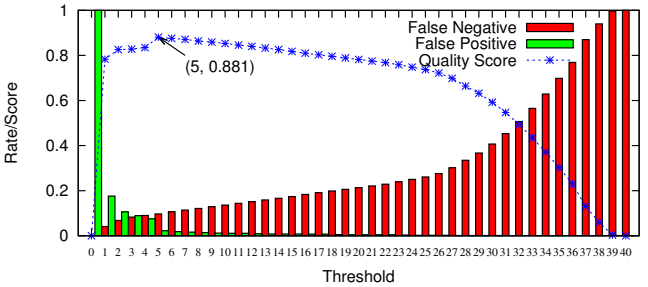

Fig. 3. TP, FP, and Quality Scores of Static Threshold-based Model with Different Thresholds (based on dataset S3, S6)

E. 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 τ_w. In our implementation, we use the quality scores computed in Section IV-C as the weight of all AVs. i.e., WA only raises a malware alarm if $\frac{\sum_{j\in\mathcal{N}_i} Q_j V_j}{|\mathcal{N}_i|} \geq \tau_w$,

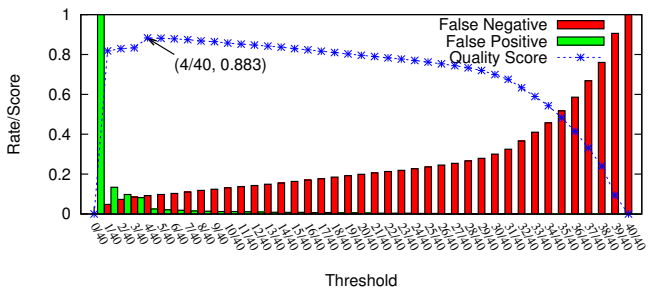

Fig. 4. TP, FP, and Quality Scores of Weighted Average Model with Different Thresholds (based on dataset S3, S6)

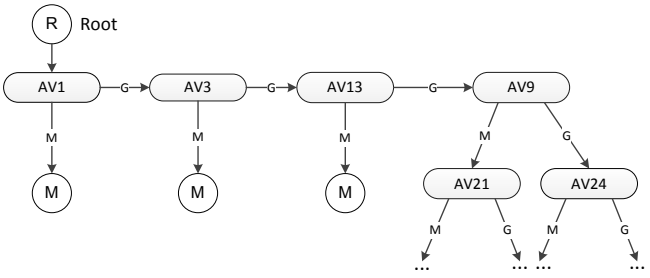

Fig. 5. The Optimal Decision Tree Generated by Weka $J48$ Algorithm (Top 5 levels)

where $V_j \in \{0, 1\}$. As shown in Fig. 4, 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 τ_w to $4/40$ unless we specify otherwise.

F. 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. Fig. 5 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 III). 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.

G. 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 III.

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

TABLE III
QUALITY SCORES AMONG DIFFERENT DECISION MODELS

Method	True Positive TP	False Negative FN	False Positive FP	Quality Score
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

H. RevMatch

The RevMatch model (Section III) 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. We fix parameters $\alpha = 1$, $\beta = 0$, and $\mathbf{P}_M = \mathbf{P}_G = 0.5$. In the first experiment, we fix parameters $C_{fp} = C_{fn} = 1$ and increase threshold τ_c from 1 to 5. As shown in Fig. 6, a higher τ_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}. Fig. 7 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.

I. 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 III. 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 the best AV.

Next, we increase C_{fn} from 1 to 13 and plot the quality score of all decision models in Fig. 8. We can see that RevMatch is superior to all others in all cases. BD performs the worst on higher 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 opinions 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}.

J. Robustness against Insider Attacks

In an open CMD, adversaries may join the collaboration and serve as co-operative CMD members in the beginning and then suddenly become malicious and send false feedback. The tasks of quickly identifying and removing malfunctioning or

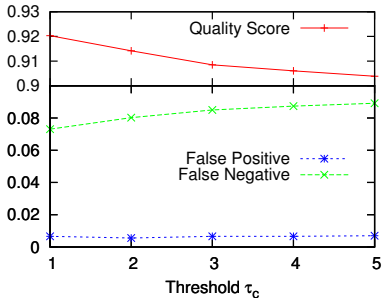

Fig. 6. The Impact from τ_c in RevMatch Model

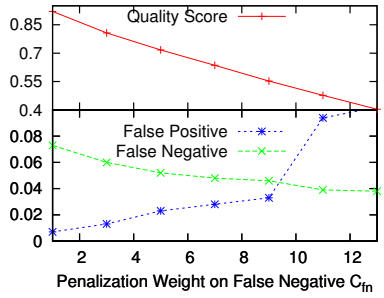

Fig. 7. The Impact from C_{fn} in RevMatch Model

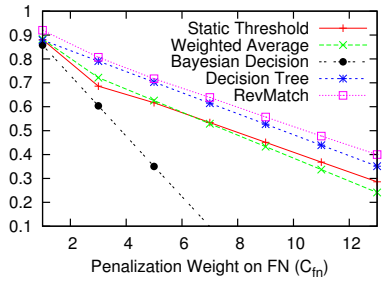

Fig. 8. Quality Scores of all Models With Different C_{fn}

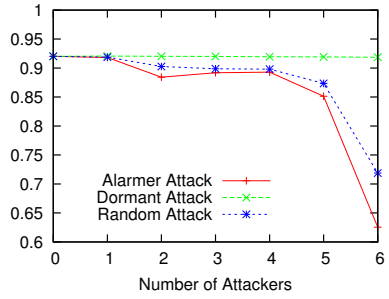

Fig. 9. RevMatch Model Under Three Different Attacks

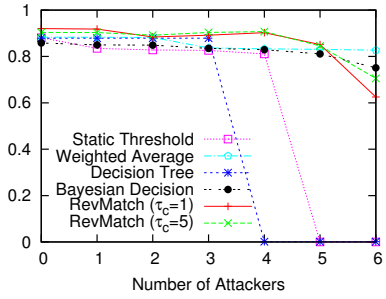

Fig. 10. The Quality Scores versus the Number of Attackers

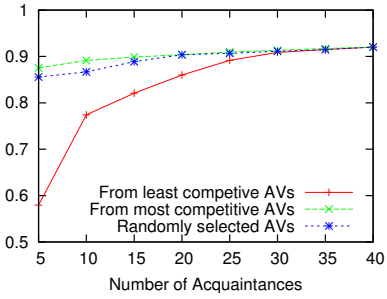

Fig. 11. The Quality Scores versus Number of collaborators

malicious insiders are the responsibilities of trust management and acquaintance management. However, in this subsection, we are interested in knowing the maximal impact malicious nodes can bring to the system if such a malicious node identification and removal mechanism does not exist. We evaluate the impact of malicious insiders on the four decision models by intentionally injecting attacks into the experimental data.

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). Fig. 9 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.

In another experiment, we investigate the impact of alarmer attacks on different decision methods. Fig. 10 shows that the decision tree is the 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 τ_c in RevMatch increases the resistance against attackers while decreasing the detection quality when there is no insider attack.

K. Acquaintance List Length and Efficiency

In the previous experiments, we showed that collaboration can effectively improve the intrusion detection accuracy for all participating AVs. In this experiment, we study the impact of collaboration network size on the overall detection quality in the network. We start with 5 AVs with the lowest ranking and gradually increase the network size by adding more competitive AVs until it reaches 40, and we observe the malware detection quality score with different network sizes. We repeat the above process by starting from the top ranking AVs and add lower ranking ones in the second experiment, and by adding randomly picked AVs in the third experiment. The results (Fig.11) show that collaboration significantly improves the detection accuracy for nodes with low detection capability and nodes with high detection accuracy also benefit from it.

We can see that although the collaboration between the top 5 AVs already yields good results, recruiting more AVs with lower ranking can further improve the overall accuracy. In all cases, a network with 25 AVs can achieve high malware detection quality. The drawback of collaborating with many AVs is the maintenance overhead since the participating AVs need to allocate resources to assist their collaborators. A host should select an appropriate acquaintance list size depending on the amount of resources it can reserve for AV collaboration.

V. Discussion

In the previous 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 use 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 CMD. They are: runtime efficiency, ability to work with partial feedback, and tuning flexibility.

A. Runtime Efficiency on Decision

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.

B. Partial Feedback

In a CMD, some collaborators may not respond to scanning requests all the time, 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.

C. 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 τ_c for the robustness of the system.

TABLE IV
Performance Summary of Collaborative Decision Models

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

D. Comparison

Table IV provides a qualitative performance comparison 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 ability to work with partial feedback. It also performs well in terms of runtime efficiency and resistance against insider attacks. Our results can be used as a reference for decision makers regarding which collaborative decision method to employ in their CMDs.

E. Zero-day attack Detection

In a CMD, behaviour-based malware detection techniques might be employed by some AV vendors. Zero-day malware can be possibly detected by some AVs who have sophisticated behaviour analysis engines. Collaboration makes it possible for AVs to exchange information on zero-day malware and thus significantly benefits users of AV products who do not have the capability to detect zero-day malware.

F. History Poison Flooding Attack

Since Revmatch uses history data for decision, adversaries may try to poison the history data to benefit themselves. For example, an adversary knows about a type of zero-day malware (it may even release it), so it always identifies this zero-day malware while the other AV engines miss it. After that, it suddenly reports all goodware to be malware, intending to cause its collaborators to raises a large number of false alarms. However, Revmatch is resistant to this type of attack since the ground truth update mechanism design prevents the adversary from poisoning the history data quickly, by using the minimum recording time gap Δt. It is difficult for the adversary to constantly create new types of zero day malware to boost its credit. Also, nodes in the network only consult others when the received file is detected as suspicious by the anomaly-based detection and a goodware usually does not raise any concern in this case.

VI. Conclusion and Future Work

In this paper, we proposed a decision model named RevMatch, which makes collaborative malware detection decisions 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 using 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. As our future work, we plan to introduce more sophisticated insider attacks, and devise corresponding defence mechanisms. We also intend to further improve the efficiency of the decision algorithm by integrating the confidence level in detection from all participating AVs.

ACKNOWLEDGMENTS

We would like to thank Jiyong Jiang from UIUC for his support and contribution on this project. We also like to thank Shameel Abdullah for his participation and for his contribution in data collection and data formatting.

REFERENCES

[1] Antivirus vendors go beyond signature-based antivirus. http://searchsecurity.techtarget.com/magazineContent/Antivirus-vendors-go-beyond-signature-based-antivirus.

[2] Bredolab Bot Herder Gets 4 Years for 30 Million Infections. http://www.wired.com/threatlevel/2012/05/bredolab-botmaster-sentenced.

[3] Evolving DDOS Attacks Provide the Driver for Financial Institutions to Enhance Response Capabilities. http://www.alston.com/Files/Publication/dc282435-c434-42a2-afe7-38af660dc82a/Presentation/PublicationAttachment/2c3bb5d8-b035-4d03-8e3c-390c2da3751d/Cyber-Alert-Evolving-DDOS-Attacks.pdf [Last accessed in April 5, 2013].

[4] McAfee antivirus to reimburse consumers for bad update. http://news.techworld.com/security/3221657/mcafee-antivirus-to-reimburse-consumers-for-bad-update.

[5] Protecting against the rampant Conficker worm. http://www.pcworld.com/article/157876/protecting_against_the_rampant_conficker_worm.html.

[6] Trend glitch costs 8 million. http://news.cnet.com /Trend-glitch-costs-8-million/2110-1002_3-5789129.html.

[7] M. Cai, K. Hwang, Y. Kwok, S. Song, and Y. Chen. Collaborative internet worm containment. *IEEE Security & Privacy*, 3(3):25–33, 2005.

[8] D. Chau, C. Nachenberg, J. Wilhelm, A. Wright, and C. Faloutsos. Polonium: Tera-scale graph mining and inference for malware detection. In *Proccedings of SIAM International Conference on Data Mining (SDM)*, 2011.

[9] 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.

[10] 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.

[11] M. Fredrikson, S. Jha, M. Christodorescu, R. Sailer, and X. Yan. Synthesizing near-optimal malware specifications from suspicious behaviors. In *Security and Privacy (S&P), 2010 IEEE Symposium on*, pages 45–60. IEEE, 2010.

[12] C. Fung and R. Boutaba. Design and management of collaborative intrusion detection networks. In *15th IFIP/IEEE Intl. Symp. on Integrated Network Management, dissertation track*, 2013.

[13] C. 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.

[14] J. Jang, D. Brumley, and S. 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.

[15] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang. Effective and efficient malware detection at the end host. In *Proceedings of the 18th conference on USENIX security symposium*, pages 351–366. USENIX Association, 2009.

[16] D. Komashinskiy and I. 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.

[17] M. Marchetti, M. Messori, and M. Colajanni. Peer-to-peer architecture for collaborative intrusion and malware detection on a large scale. *Information Security*, pages 475–490, 2009.

[18] C. A. Martínez, G. I. Echeverri, and A. G. C. 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.

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

[20] C. Silva, P. Sousa, and P. Verissimo. Rave: Replicated antivirus engine. In *Dependable Systems and Networks Workshops (DSN-W), 2010 International Conference on*, pages 170–175. IEEE, 2010.