Computer Networks 169 (2020) 107027

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Maximizing accuracy in multi-scanner malware detection systems

Muhammad N. Sakib[a], Chin-Tser Huang[a,*], Ying-Dar Lin[b]

[a] Department of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA
[b] Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan

ARTICLE INFO

Article history:
Received 4 March 2019
Revised 2 October 2019
Accepted 22 November 2019
Available online 30 November 2019

Keywords:
Multi-scanner
Modeling
Malware detection
Accuracy

ABSTRACT

A variety of anti-malware scanners have been developed for malware detection. Previous research has indicated that combining multiple different scanners can achieve better result compared to any single scanner. However, given the diversity in detection rates and accuracy of different anti-malware scanners, how to determine the best possible outcome of multi-scanner systems in terms of accuracy and how to achieve this best outcome remain formidable tasks. In this paper, we propose three models to capture the combined output of different combinations of anti-malware scanners based on the limited amount of historical information available. These models enable us to predict the accuracy level of each combination, which helps us to determine the optimal configuration of the multi-scanner detection system to achieve maximum accuracy. We also introduce two methods to identify a near-optimal subset of scanners that can help reduce scanning cost while under time constraint. From simulations over randomly generated hypothetical datasets and experiments conducted with real world malware and goodware datasets and anti-virus scanners, we found that our models perform well in predicting the optimal configuration and can achieve an accuracy as high as within 1% of true maximum.

1. Introduction

Malicious software (malware) is one of the major tools of cybercriminals. Almost every cyber-attack involves some kind of malware as the facilitator. Therefore, detection of malware is one of the core problems of modern cyber security. For a long time, we have been relying on the anti-malware or anti-virus scanners to detect malware and to protect our systems from malware-associated attacks. A variety of anti-malware scanners have been developed over the years with different levels of performances. In the early days, a single scanner was sufficient and able to detect most of the malware out there. However, as time goes on, the malware writers have honed their skills and their repository of malware has evolved and proliferated so much that no single anti-malware engine can protect us from all of them. Moreover, researchers have determined that combining the power of multiple anti-malware engines improves detection accuracy and performance significantly compared to any single anti-malware scanner (for example, [1,2]). Consequentially, various online multi-AV scanning services and tools (VirusTotal [11], Jotti [12], VirScan [13], etc.) have been developed for addressing this concern.

Although we have several multi-AV scanning services and tools at our disposal, most of them are used only for informational purposes or as a source of second opinion. None of them directly provide an exact decision of whether a given sample is malicious or benign. Instead, they work as an information aggregator and only list the individual results returned from each anti-virus scanning engine. The responsibility of making a decision based on these individual scan results is up to the human user. This may be convenient for personal use where an end-user is looking for a second opinion for an unknown sample downloaded from the Internet. However, if we want to use these multi-scanner detection systems effectively for a large-scale detection and collection operation, we need the system to automatically come up with an accurate decision. The importance of this decision making does not end with the individual end-users. As pointed out by the researchers in [35,38], the impact of the lack of consensus, consistency and correctness among many available anti-virus scanners' decisions is wide ranging, affecting not only academic researchers and anti-virus scanner vendors, but also IT professionals who manage security operations in enterprise networks, and government agencies in charge of protecting critical infrastructure from cyber-attacks. Therefore, it is imperative to construct a multi-scanner system with accurate decision-making capabilities that will enable both the generation of authoritative ground truth datasets and the development of real time attack response systems against modern network-based attacks.

* Corresponding author.
E-mail addresses: sakib@email.sc.edu (M.N. Sakib), huangct@cse.sc.edu (C.-T. Huang), ydlin@cs.nctu.edu.tw (Y.-D. Lin).

https://doi.org/10.1016/j.comnet.2019.107027
1389-1286/© 2019 Elsevier B.V. All rights reserved.

To come up with accurate decisions, a multi-scanner system would need enormous information about the dataset for training and calculation purposes. However, there are only two types of limited information available when the multi-scanner system faces an unknown sample set, and they are not guaranteed to be reliable for the following reasons. The first type of available information is the individual scan results from various scanners, but they at best can only be considered as the scanners' "opinions" because we don't know for sure whether they are right or wrong. The second type is the statistics for each scanner indicating their accuracy and performance. These statistics are accumulated from previous scanning results which can be proven as right or wrong over the course of time with or without the help of manually vetted ground truth datasets. These statistical accuracy values can be used to measure how right or wrong these scanners can be. In other words, these are the ratings that indicate how well these scanners performed. With only such limited information available, the original problem now becomes how to combine the previous detection accuracy statistics for each scanner and the actual scan results for a given unknown sample to classify the sample as benign or malicious with the best possible accuracy.

In this paper, we tried to solve this problem by first deriving a mathematical model named Combined Probability Model (CPM) to capture the combined outcome of a specific combination of scanners, given only their individual detection rates. That means, no information about the interdependency between each pair of two scanners is available, and therefore, we develop the model based on the assumption that the scanners are completely independent. This mathematical model consists of a set of formulas involving individual detection probabilities of the scanners. Next, we assume that the pairwise dependency information among the scanners is available, which means we have the combined probability of detection for all possible pairs of scanners. This dependency among the scanners are due to the fact that many features that the scanners use to classify malware are common among different scanners (for example, hash of malicious binary, pattern of instruction sequences, etc.). Using this dependency information, we developed a Dependency Approximation Model (DAM) to calculate combined probability of scanner sets with size 3 and above. Since we have some dependency information about the scanners, we don't need the assumption of independence for this model. While these first two models can have good accuracy in calculating combined probabilities, they have exponential runtime which could become a problem for a large set of scanners. Therefore, we developed a greedy heuristic based approximation models called Greedy Approximation Model (GAM). This model applies greedy approximation over CPM formulas to improve runtime and at the same time try to maintain the accuracy as much as possible. These models give us a good approximation of the combined true and false detection probabilities of the combined system of scanners, which can be used to calculate the overall accuracy of the multi-scanner system for a specific configuration. Therefore, if we can calculate the accuracy of all possible configurations of the system, we can compare them to determine the optimal configuration that provides us with the maximum accuracy.

In addition to the original problem, we also try to answer the following two questions: (1) Is it always beneficial to increase the number of scanners in a multi-scanner detection system? (2) How can we select a subset from all available scanners, which will provide us with a maximum accuracy for a size of the given subset? To address the second question, we come up with two methods – the ranking method and the dependency approximation method, which allow us to select a best subset from the full set of scanners. To verify the accuracy of our models and to answer these additional questions, we first numerically simulate our models over randomly generated hypothetical datasets and test case scenarios. From the simulation results, we found that if the average false positive rate of the scanners is high enough, the accuracy value of multi-scanner system can decrease at some point with the increase in the number of scanners. At the end, we provide experimental evaluation based on real-world malware and goodware ground truth datasets and corresponding anti-virus scanning results using a popular online multi-AV scanning service, VirusTotal. From the evaluations, we can verify the accuracy of our simulation results and establish that our models along with the methods for finding best subset of scanners perform extremely well in predicting the optimal configuration to achieve a maximum accuracy based on available information, which is within 1% of the true maximum.

The remainder of this paper is organized as follows. Section 2 discusses previous work done on multi-scanner architectures and collaborative malware detection architectures. Section 3 presents the formulation of the problem. Section 4 introduces Combined Probability Model along with the mathematical derivation of the formulas. Section 5 describes the Dependency Approximation Model. Section 6 describes the greedy heuristic based model. Section 7 provides a background on the evaluation metrics used to calculate accuracy and describes our methods to find the best subset. Section 8 presents numerical evaluation results based on the proposed models and methods. Section 9 presents the experimental evaluation of the models and the methods based on real-world datasets and VirusTotal scanning reports. Finally, in Section 10 we provide conclusions and lessons learned from our models and experimental results including possible future work.

2. Related work

In this section, we have briefly described and discussed previous and existing related research and commercially available products involving multi-scanner systems. This includes research on multi-scanner model and architecture, collaborative malware detection, existing multi-AV scanning services and software, and commercially available anti-virus scanners with multiple built-in scanning engines.

2.1. Multi-scanner model and architecture

Morales et al. [1] experimentally showed that a single anti-malware program is not sufficient to detect all malware present on a system. Though in a limited fashion, their results showed that combining multiple anti-malware programs achieves better recall and false negative rates. Cukier et al. [2] presented empirical evidence that detection capabilities are considerably improved by diversity with AVs, and their findings also showed that none of the single anti-virus software achieved perfect detection rate. These two works ([1,2]) mainly validate the importance of combining multiple scanners instead of investigating how to do it to achieve maximum accuracy. Oberheide et al. [3] presented a new model for malware detection on end hosts based on providing anti-virus as an in-cloud network service. Their model used multiple, heterogeneous detection engines in parallel, a technique termed as "N-version protection". To verify their model, they constructed and deployed an in-cloud antivirus system called CloudAV. CloudAV includes a lightweight, cross-platform host agent and a network service with ten anti-virus engines and two behavioral detection engines. Their experimental results showed that CloudAV provides 35% better detection coverage against recent threats compared to a single anti-virus engine and a 98% detection rate across the full dataset. The similarity between CloudAV and our proposed multi-scanner system is only in the architecture of the system. The result aggregation in CloudAV is done in a very inefficient way, where a manually defined security policy is used to decide on a threshold value at which a candidate file is deemed unsafe or malicious. CloudAV has no automated result aggregation or decision-making process in place to dynamically select the optimum threshold to reduce false positives or false negatives; in comparison, we propose three models to solve this problem. Chiriac [28] presented a comparative analysis of cloud-based scanning and local scanning, and proposed a trade-off to make use of the best parts of both. Sebastián et al. presented AVclass [32], an automatic labeling tool that uses the AV labels assigned by multiple AV engines for a massive number of samples to generate the most likely family names for each sample, ranking each potential family name by the number of AV engines that assign this name to the sample. Both of these works [28,32] are fundamentally different compared to our work, although the insights from our work could benefit their work in many ways.

The problem of finding the ground truth decision label has been the focus of several recent research work done in the malware detection area concerning multiple anti-virus scanners. Mohaisen and Alrawi in AV-Meter [35] demonstrate the danger of relying on incomplete, inconsistent, and incorrect malware labels provided by anti-virus vendors. Their study showed that we need many independent anti-virus scanners to obtain complete and correct labels, while it is sometimes impossible to achieve such goal using multiple scanners. Kantchelian et al. [27] examined the problem of aggregating the results of multiple anti-virus (AV) vendors' detectors into a single authoritative ground-truth label for every binary file, and proposed a machine learning based solution using both unsupervised and supervised techniques. This work has a major difference compared to ours in that it requires a comprehensive training dataset comprising of same anti-virus scanners and ground truth labels for the models to be effective, whereas our models only require detection probability values to be available for the anti-virus scanners regardless of historical datasets being used to calculate them. In addition, their models assume independence across anti-virus scanners and datasets, whereas our DAM model incorporates dependency information of the scanners and does not require the independence assumption. Furthermore, they did not address additional questions regarding the size of the scanner set and how to select them. Similarly, Charlton et al. [37] proposed an algorithm to estimate the relative accuracy of the malware detectors in the absence of ground truth of their actual detection quality. Hurier et al. [38] explored the lack of agreement among AV engines and proposed a set of metrics that quantitatively measure different dimensions of this lack of consensus.

There have been numerous research papers on combining multiple classifiers to achieve highest classification accuracy, which might deceptively seem to be related to the problem of combining multiple anti-virus scanners to achieve maximum accuracy. For example, dynamic weighted voting scheme proposed by Valdovinos and Sanchez [33] and regression methods proposed by Górecki and Krzyśko [34] are representative research works on combining multiple classifiers. However, we have identified several fundamental differences between the problem of combining multiple classifiers and the problem of combining multiple anti-virus scanners, which make existing solutions available for the former problem unsuitable for the latter problem. We list these key differences as follows.

1. Classifiers are completely based on training data. Although anti-virus scanners might also include training data or detection history, they are not completely dependent on them. Many scanners apply heuristic rules or other static or dynamic analysis methods.
2. We know the underlying methodology of most well-known classifiers. However, most commercial anti-virus scanners hide their underlying proprietary methodology. In our proposed multi-scanner modeling scheme, we treat the scanners as black boxes.
3. Classifiers assign a classification score to input samples based on the features and apply a class label according to this score. All of these are known. On the other hand, since we don't know the internal methodology of most anti-virus scanners, we only get the output label from a scanner, which is most of the time either malicious or benign along with a label identifying the malware type or family if the sample is malicious. We don't know the classification score, if there is any, from most scanners.

In summary, we can see that the differences between combining classifiers and combining anti-virus scanners lie in the lack of necessary information needed by existing multi-classifier models. In this paper, we aim to find a solution to the problem of combining multiple anti-virus scanners to achieve maximum possible accuracy using available information which only includes a detection probability for each scanner which is calculated from the scanner's past detection results, pairwise detection probability (if available) to account for scanner dependency and the current detection result. It does not require a classification score or measurement of each scanner's level of confidence in determining the class or label for an input sample, which is vital for combining multiple classifier systems. For this reason, we neither include a thorough study of related research done on combining multiple classifiers nor compare these works with our proposed models.

To provide a good understanding of machine learning based approaches on the malware detection problem, here we present a brief discussion on them. The vast amount of existing research on employing machine learning for malware detection are mainly based on three different types of malware analysis [51] – static, dynamic and hybrid analysis. The static analysis approaches are based on static features of malware such as extracted opcode sequences, function call graphs, control flow graphs, etc., in which researchers have applied hidden Markov models [52], profile hidden Markov models [53], support vector machine [54] principal component analysis [55,56], clustering [57], etc. to mention a few from the extensive research in this area. In contrast, the dynamic analysis approaches focus on malware behavior and interactions captured via sandboxing, controlled execution, network traces, etc. Researchers have used clustering [39,40,42,46,50], classification via support vector machine, etc. [39,43], deep learning [41,47,48], family classification [49,50], multiple classifiers [44], multiple instance learning [45] to analyze these dynamic contexts. The hybrid analysis approaches combine various aspects of static and dynamic analysis with the hope to benefit from the advantages of both categories, in which researchers used machine learning techniques such as those in [58] and [59] to classify malware. Apart from these, there are many existing research works that combine different machine-learning or classifier based techniques to improve detection of malware. Examples of such work include [60] where the authors use support vector machines to combine scores from three advanced malware scoring techniques to obtain improved results compared to using individual scores. Similar other works include [61,62], and [63]. A fundamental difference between these existing machine learning or classifier based works and our proposed method is that these works consider and use the features (both from static and dynamic analysis) directly extracted from the malware, whereas our method only considers and uses the detection rates of the scanners and does not consider or use the features directly extracted from the malware at all. In addition, we have already mentioned the key differences between combining multiple classifiers and combining multiple anti-virus scanners, which is the way we propose in this paper.

2.2. Collaborative malware detection

There has been some research on the collaborative approach in detecting malware. Schmidt et al. [4] presented a collaborative malware detection approach to reduce false negative rate for Android-based malware detection by performing static analysis of executables and sharing detection information among neighboring nodes. Fung et al. [5] presented a collaborative decision-making approach for malware detection systems. They proposed a decision model called RevMatch [6], where collaborative malware detection decisions are made based on the scanning history with multiple anti-virus systems. They claimed that the experimental evaluation of their model shows significant improvement over any single anti-virus engine. RAVE [7] is a centralized collaborative malware scanning system for email infrastructures where email correspondence is used to contact multiple agents for malware scanning and a voting mechanism is used to make the final decisions. Marchetti et al. [8] presented a distributed peer-to-peer architecture for collaborative malware and intrusion detection focusing more on dependability and load-balancing issues. A similar approach was proposed by Colajanni et al. [9]. Lu et al. [10] presented SCMA, a distributed malware analysis system with the goal of better collaboration and scalability. Aldini et al. [30] presented a collaborative approach to detect repackaged mobile applications. Miller et al. [36] investigated the effect of integrating expert reviewers into a scalable malware detection system and demonstrated that even in small numbers, reviewers can vastly improve the system's ability to keep pace with evolving threats. Each of these collaborative approaches require additional internal information from the anti-virus scanners as well as active collaboration from them, which makes these works different from our work where we make use of the anti-virus scanners as static functional blocks with no knowledge of internal mechanism.

2.3. Multi-AV scanning services and software

There are free online public services that provide scanning reports from multiple anti-virus scanners. VirusTotal [11], a Google subsidiary, is the most prominent among these services. VirusTotal uses the command-line versions of 59 anti-virus scanners (at the time of writing) to scan a single file and include the results returned by each scanner into an aggregated report. In addition to telling whether a given anti-virus solution detected a submitted file, it displays the exact detection label returned by each engine. This service is mainly useful to the anti-virus vendors and to those private users who want a second opinion. Among other such services, there are Jotti [12], VirSCAN [13], File2Scan [14], and Metadefender [15], where File2Scan and Metadefender are paid services. There are also multi-AV scanning client tools such as HerdProtect [16], HitmanPro [17], SecureAPlus [18], and Multi-AV [19]. Our work is intended for standalone multi-scanner detection systems that operate in the same way as these existing multi-scanning services and software and also incorporate an automated decision-making process which will yield maximum accuracy in malware detection.

Researchers have done research analyzing VirusTotal data as well. Algaith et al. [29] compared the detection capabilities of the version of nine AV products that the vendors make available for free in VirusTotal versus their full capability versions that they make available via their own website and found that only one of the vendors had a full capability version which detected all the malware that their VirusTotal version could detect. Song et al. [31] performed a large-scale examination of VirusTotal repository and their results show that malwares appear in bursts and that distributions of malwares are highly skewed.

2.4. Commercial AV scanners with multiple scanning engines

Most of the anti-virus vendors use their own proprietary malware detection engine which usually includes a signature database, a heuristic-based detection engine, and a reputation-based detection system. A few of them, namely Emsisoft [20] and G Data [21], use a dual-engine technology where each scan passes through two engines, but the benefit of this dual engine technology and how it works has not been made publicly available. Therefore, it is safe to assume they have not incorporated any extensive multi-scanning technology in their products.

From the above discussion on related research on multi-scanner systems, we observe that although a few of them tackled the same problem space as ours, most of the previous works were fundamentally different in their focus and approach and limited in their contribution in solving the problem of configuring a multi-scanner system. In comparison, in this paper we address some fundamental questions regarding multi-scanner systems as outlined in the next section and propose solutions and provide guidelines and insights to solve them.

3. Problem formulation

In this section, we present our formulation of the problem of maximizing accuracy in a multi-scanner detection system using appropriate formal notations. Table 1 lists some of these notations used in the formulation. Formally, the problem of maximizing accuracy in a multi-scanner detection system can be stated as follows:

Given N scanners along with their respective (true positive and false positive) detection rates or probabilities P_i (where $1 \le i \le N$), and the binary detection results (either true or false) for a given sample obtained from these N scanners, how can we find the optimal value of Th ($1 \le T \le N$) where Th is the threshold to decide the maliciousness of that given sample. Here, we assume that N is a finite number and we only have the detection rates or probabilities associated with each scanner that can be calculated from past detection results of the scanners provided the detection results accurately reflect the decisions generated by the scanners.

The problem can be extended further to answer the following questions:

1. Assuming that N is the total number of scanners that we can use, and Q is the optimal number of scanners to achieve maximum accuracy, what is the relationship between N and Q? Does $Q = N$ always hold, or can $Q < N$ be true in some cases? In other words, does adding another scanner always improve accuracy?
2. If M is the size of a subset of all N scanners, how do we select these M scanners from all N scanners to achieve maximum accuracy that is possible for any subset of scanners of size M.

Table 1
Notations.

Symbol	Description
I	Input to the multi-scanner system
O_i	Output of ith scanner (0 or 1)
N	Total number of available scanners
Q	Optimal number of scanners to achieve maximum accuracy
Th	Threshold to decide the maliciousness of an object
P_i	Detection probability of ith scanner
${P^T}_i$	The probability of classifying a malicious object as malicious by ith scanner
${P^F}_i$	The probability of classifying a benign object as malicious by ith scanner
$CP(t)$	Combined detection probability when $Th = t$

In other words, given that there can be $\binom{N}{M}$ possible subsets of size *M*, which subset is the best subset of *M* scanners such that it will provide maximum accuracy among all these possible subsets?

4. Combined Probability Model (CPM)

In this section, we will explain the development of the *Combined Probability Model (CPM)* in detail. As mentioned earlier, we have devised a set of formulas to construct the model. In the formulas, we used certain symbols and notations to denote various terms. Table 1 lists these notations. To help the readers better understand the model, we will start with a small-scale model consisting only three scanners. Then, we will extend the small-scale model to a more generalized version.

4.1. Three-scanner CPM

We start with a simple three-scanner model ($N = 3$) to explain the method of developing the generalized model. The most generic multi-scanner system consisting three scanners is depicted in Fig. 1. We assume here that all the scanners are binary scanners, i.e., they produce an output of either 1 or 0, where a 1-output means that the sample is detected as malicious and 0-output means that the sample is detected as benign. We further assume that the detection probabilities for each scanner is given or calculated using a diverse and extremely large sample set with reasonable accuracy to prevent a significant bias towards a particular class of samples. More importantly, we assume that these detection probabilities are independent of each other, although in reality they are not. This assumption is simply because of the unavailability of such dependency information. For the case of limited availability of such information, we designed the Dependency Approximation Model (DAM) (described in Section 5) which does not require this independence assumption.

In the three-scanner model, we have three possible choices of threshold *Th* (3, 2 and 1) to decide maliciousness of an input sample. We derive the equations of our model by adding up the smaller components of the probabilities. For example, for $Th = 1$, the combined detection probability is specified as

$$CP(1) = P\{X \geq 1\} = P\{X = 1\} + P\{X = 2\} + P\{X = 3\}$$

where *X* denote the random variable defined as the number of scanners that detect a given sample as malicious. Therefore, we generalize this equation for $Th = t$ (where $1 \leq t \leq 3$) as

$$CP(t) = \sum_{i=t}^{3} P\{X = i\}. \tag{1}$$

P_1, P_2, and P_3 denote individual detection probabilities for *scanner 1*, *scanner 2*, and *scanner 3* respectively. According to the rules of probabilities (assuming independence) we can write

$$\begin{aligned} P\{X = 1\} = {} & P_1(1 - P_2)(1 - P_3) \\ & + P_2(1 - P_3)(1 - P_1) \\ & + P_3(1 - P_1)(1 - P_2), \end{aligned} \tag{2}$$

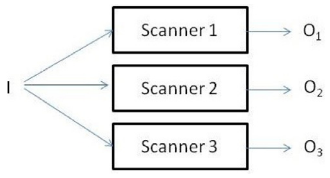


Fig. 1. A 3-scanner system.

$$\begin{aligned} P\{X = 2\} = {} & P_1P_2(1 - P_3) \\ & + P_2P_3(1 - P_1) \\ & + P_3P_1(1 - P_2), \end{aligned} \tag{3}$$

and

$$P\{X = 3\} = P_1P_2P_3. \tag{4}$$

The reasoning behind these equations is also illustrated in Fig. 2. Replacing the values from Eqs. (2)–(4) into Eq. (1), we can easily calculate the combined probability (*CP*) for a given $Th = t$.

4.2. N-scanner CPM

In Section 4.1, we limited our discussion to only three scanners for ease of understanding. Next, we extend this three-scanner model to a general *N*-scanner model. Fig. 3 shows an *N*-scanner system.

For an *N*-scanner model with $Th = t$ (where $1 \leq t \leq N$), Eq. (1) becomes

$$CP(t) = \sum_{i=t}^{N} P\{X = i\}. \tag{5}$$

Based on Eqs. (2)–(4), we can come up with a generalized *N*-scanner equation for the probability $P\{X = i\}$ as

$$P\{X = i\} = \sum_{j=1}^{\binom{N}{i}} \prod_{k=1}^{i} P_k^j \prod_{l=i+1}^{N} (1 - P_l^j), \tag{6}$$

where P_k^j is the probability of the scanner with index k ($1 \leq k \leq i$) in *j*th combination in $\binom{N}{i}$ and P_l^j is the probability of the scanner with index l ($i + +1 \leq l \leq N$) in all the other scanners that are not in *j*th combination. Substituting the value of $P\{X = i\}$ from Eq. (6) into Eq. (5) we get

$$CP(t) = \sum_{i=t}^{N} \sum_{j=1}^{\binom{N}{i}} \prod_{k=1}^{i} P_k^j \prod_{l=i+1}^{N} (1 - P_l^j). \tag{7}$$

Eq. (7) can be used as the generic *N*-scanner equation for combined detection probability when $Th = t$.

5. Dependency Approximation Model (DAM)

In this section, we present the *Dependency Approximation Model (DAM)* where we have dependency information among all pairs of scanners in the set of scanners. Let's start with a three-scanner model, as depicted in Fig. 4. We assume that we have $P_1 \cap P_2$, $P_2 \cap P_3$ and $P_3 \cap P_1$ available along with P_1, P_2 and P_3. Our goal is to calculate an estimated value of $P_1 \cap P_2 \cap P_3$. We observe that all three of $P_1 \cap P_2$, $P_2 \cap P_3$ and $P_3 \cap P_1$ have $P_1 \cap P_2 \cap P_3$ in common. So, our idea is to estimate the portions from each of $P_1 \cap P_2$, $P_2 \cap P_3$ and $P_3 \cap P_1$ that contributes to $P_1 \cap P_2 \cap P_3$ and take the average value of them. To calculate this, we notice that the following fractions are approximately proportional to each other:

$$\frac{P_1 \cap P_2 \cap P_3}{P_1 \cap P_2} \approx \frac{P_3 \cap P_1}{P_1}$$

implies

$$P_1 \cap P_2 \cap P_3 \approx P_1 \cap P_2 \times \frac{P_3 \cap P_1}{P_1}$$

Similarly, since

$$\frac{P_1 \cap P_2 \cap P_3}{P_1 \cap P_2} \approx \frac{P_2 \cap P_3}{P_2}$$

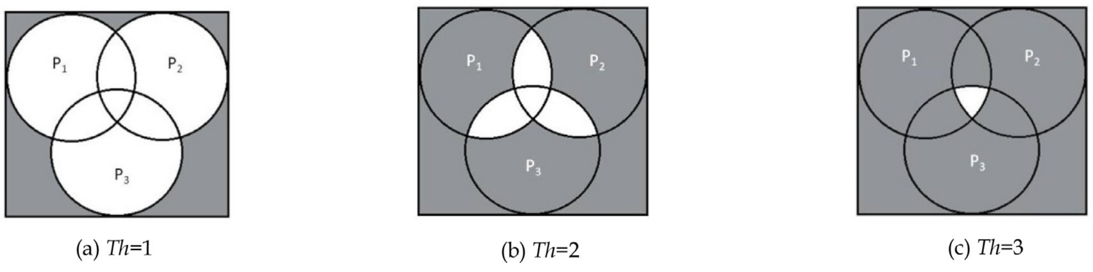


Fig. 2. Venn diagrams for the three cases.

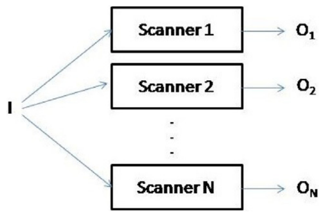


Fig. 3. An *N*-scanner system.

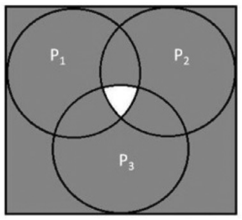


Fig. 4. A 3-scanner system, white region shows $P_1 \cap P_2 \cap P_3$.

Algorithm 1 Dependencyapproximation(L_1, L_2).

```
1: $L_{3\text{-}N} \leftarrow$ [] //initialize list $L_{3\text{-}N}$
2: for $i$ in range(3, $N$) do
3:     $L_{comb(i)} \leftarrow$ ListCombination($L_N$, $i$)
4:     for each $comb_i$ in $L_{comb(i)}$ do
5:         $L_{sum} \leftarrow$ [] //initialize list to calculate mean
6:         $L_{comb(i\text{-}1)} \leftarrow$ ListCombination($comb_i$, $i$-1)
7:         for each $comb_{i\text{-}1}$ in $L_{comb(i\text{-}1)}$ do
8:             $CP_{comb(i\text{-}1)} \leftarrow$ FindValue($L_{3\text{-}N}$, $comb_{i\text{-}1}$)
9:             $r \leftarrow$ RemainingScanner($comb_i$, $comb_{i\text{-}1}$)
10:            $P_r \leftarrow$ FindValue($L_1$, $r$)
11:            for each $j$ in Index($comb_{i\text{-}1}$) do
12:                $CP_2 \leftarrow$ FindValue($L_2$, $j$, $r$)
13:                $res \leftarrow CP_{comb(i\text{-}1)} \times CP_2 \,/\, P_r$
14:                $L_{sum}$.append($res$)
15:                $CP_{comb(i)} \leftarrow$ Mean($L_{sum}$)
16:                $L_{3\text{-}N}$.append($comb_i$, $CP_{comb(i)}$)
17:            end for
18:        end for
19:    end for
20: end for
21: return $L_{3\text{-}N}$
```

implies

$$P_1 \cap P_2 \cap P_3 \approx P_1 \cap P_2 \times \frac{P_2 \cap P_3}{P_2}$$

In this way, we can calculate four other proportional values for $P_1 \cap P_2 \cap P_3$ which along with the previous two values can be used to calculate the estimated value of $P_1 \cap P_2 \cap P_3$ as follows:

$$\bigcap_{i=1}^{3} P_i \approx MEAN\begin{pmatrix} P_1 \cap P_2 \times \frac{P_3 \cap P_1}{P_1}, & P_1 \cap P_2 \times \frac{P_2 \cap P_3}{P_2}, \\ P_2 \cap P_3 \times \frac{P_1 \cap P_2}{P_2}, & P_2 \cap P_3 \times \frac{P_3 \cap P_1}{P_3}, \\ P_3 \cap P_1 \times \frac{P_1 \cap P_2}{P_1}, & P_3 \cap P_1 \times \frac{P_2 \cap P_3}{P_3} \end{pmatrix}$$

where *MEAN* is a function that takes the average of the parameters passed to it.

We can further extend this for *N* scanners as follows:

$$\bigcap_{i=1}^{N} P_i \approx MEAN\begin{pmatrix} P_1 \cap P_2 \cap P_3 \cap \ldots \cap P_{N-1} \times \frac{P_N \cap P_1}{P_1}, \\ P_1 \cap P_2 \cap P_3 \cap \ldots \cap P_{N-1} \times \frac{P_N \cap P_2}{P_2}, \\ P_1 \cap P_2 \cap P_3 \cap \ldots \cap P_{N-1} \times \frac{P_N \cap P_3}{P_3}, \\ \ldots \end{pmatrix}$$

The recurrent nature in this formula can be used to develop a dynamic programming algorithm which calculates all the values for *N* scanners in a bottom-up fashion. Algorithm 1 shows the Dependency Approximation algorithm to calculate the table of values of all possible combinations of *N* scanners. The input to this algorithm is all the individual and pairwise detection probabilities (L_1, L_2) of the scanners. Here is a brief explanation of how the algorithm works: in line 1, we initialize the list $L_{3\text{-}N}$ for all combinations of size 3 to *N*. In line 3, we get the list of combinations $L_{comb(i)}$ for size *i* using the ListCombination function that returns a list of combinations with size *i* from a list of numbers (L_N). In line 6, the ListCombination function is similarly used. In line 8, we get the combined probability value $\text{CP}_{\text{comb(i-1)}}$ by using the FindValue function that takes a combination of scanners ($comb_{i\text{-}1}$) and finds the combined probability value from $L_{3\text{-}N}$. In line 9, we identify the remaining scanner index *r* using RemainingScanner function from $comb_i$ and $comb_{i\text{-}1}$. In line 10, we get the detection probability value for scanner index *r* from L_1 using FindValue. In line 11, we used the Index($comb_{i\text{-}1}$) function to get the all the scanner index values from $comb_{i\text{-}1}$. Again, in line 12, we used FindValue function to find the combined probability value CP_2 from L_2 using scanners *j* and *r*. Lines 13 to 16 follows the above dependency approximation formula to calculate combined probability value $CP_{comb(i)}$ and append it to list $L_{3\text{-}N}$. After the complete table of values is calculated, a top-down update of the values is used to calculate combined probability *CP(t)* for threshold *t* (CP_t in the algorithm), as shown in Algorithm 2. This is necessary, because the calculated table in Algorithm 1 contains values for $CP_{comb(i)}$ which means values for each combination of scanners with size *i*. That means, there are

Algorithm 2 CombinedProbabilityCalculation($L_{1\text{-}N}$, t).

```
1: $CP_t \leftarrow 0$ //initialize combined probability $CP_t$
2: for $i$ in range(size($L_{1\text{-}N}$) −1, 0) do
3:     $val \leftarrow$ ImmediateSuperSet($L_{1\text{-}N}[i]$)
4:     $L_{1\text{-}N}[i] \leftarrow L_{1\text{-}N}[i] - val$ //update table values
5: end for
6: for $j$ in range($N$, $t$) do
7:     $CP_t \leftarrow CP_t$+ + CalculateSum($L_{1\text{-}N}$, $j$)
8: end for
9: return $CP_t$
```

overlaps in values for scanners with size i. We need to remove this overlap to calculate an accurate sum of probability values for scanner combinations with size greater than or equal to threshold t. This is done in lines 2 to 5 in Algorithm 2, where the ImmediateSuperSet function returns the immediate super set with size $i + 1$ that contains all the combinations with size i.

6. Greedy Approximation Model (GAM)

Our models do not assume any upper bound on the number of scanners. Popular online multi-AV scanning service VirusTotal now consists of more than 70 anti-virus scanners and it is highly likely that this number of available anti-virus scanners will continue to increase. Therefore, due to the exponential run-time complexity requirement of the previous two models, we introduce a third model called the *Greedy Approximation Model* (*GAM*), which applies the greedy heuristic to approximately calculate the combined probability $CP(t)$ for a given threshold t. Here, the greedy heuristic is to start by combining the highest t individual detection probabilities and moving along in a decreasing order doing the same until less than t probabilities are available. An example would better illustrate this approach. Let's say we have P_1, P_2, P_3, ..., P_N individual detection probabilities available sorted in a decreasing order, that is, $P_1 \geq P_2 \geq P_3 \geq \ldots \geq P_N$. To calculate $CP(t)$, we initialize $CP(t)$ to 0 and calculate $P_1 \times P_2 \times P_3 \times \ldots \times P_t$ and add to $CP(t)$. For the next iteration, we calculate 1 - $CP(t)$ and multiply it with $P_2 \times P_3 \times P_4 \times \ldots \times P_{t+1}$ and add the result to $CP(t)$. This goes on till we add $P_{N-t+1} \times P_{N-t+2} \times P_{N-t+3} \times \ldots \times P_N \times (1 - CP(t))$ to $CP(t)$. The final value of $CP(t)$ is our desired combined detection probability. We developed the *Greedy Approximation* algorithm based on this approach, as shown in Algorithm 3. The parameters L_p and t refer to the list of individual detection probabilities and threshold respectively, and the resulting combined probability is denoted by CP_t.

Algorithm 3 GreedyApproximation(L_p, t).

```
1: $L_p \leftarrow$ sort($L_p$) //sort list of probabilities $L_p$
2: $CP_t \leftarrow 0$ //initialize combined probability $CP_t$
3: $CP_c \leftarrow 1$
4: for $i$ in range(0, $N - t + 1$) do
5:     $m \leftarrow CP_c$
6:     for $j$ in range($i$, $i + t$) do
7:         $m \leftarrow m \times L_p[j]$
8:     end for
9:     $CP_t \leftarrow CP_t + m$
10:    $CP_c \leftarrow 1 - CP_t$
11: end for
12: return $CP_t$
```

7. Finding best subset of scanners

Using all the scanners available might not be feasible always for reasons such as high cost and exponential execution time needed. We can decide to reduce the number of scanners being used, but then comes the problem of finding the best subset of them. We have developed two methods to solve this problem – the ranking method and the dependency approximation method. To understand these methods and later the experimental evaluation, we need to first understand the accuracy metrics.

7.1. Accuracy metrics

The simplest metric is called *Accuracy* (*ACC*) or *Fraction Correct* (*FC*) [22]. It measures the fraction of all instances that are correctly categorized and is defined by

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

where *TP*, *TN*, *FP*, and *FN* refer to true positive, true negative, false positive, and false negative respectively. However, since *TP* and *FN* should add up to the total number of positive instances (*P*), and *TN* and *FP* should add up to the total number of negative (*N*), we can calculate *FN* and *TN* from *TP* and *FP* as follows:

$$FN = P - TP$$
$$TN = N - FP$$

Another useful metric is the *F1 score* [23]. It considers both precision and recall rates of the test to compute the score and is defined by

$$F1 = \frac{2TP}{2TP + FP + FN}.$$

A third metric, called the *Matthews Correlation Coefficient (MCC)* [24], is used in machine learning as measure of quality of binary classifications. It is generally regarded as a balanced measure and is defined by

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}.$$

7.2. The ranking method

To identify the best subset of scanners for a given size M out of N ($1 \leq M \leq N$), we need to rank the scanners based on a suitable criterion that can help in achieving the maximum accuracy and select the top M scanners. But the only information about the scanners is their detection rates. Therefore, we need to create an individual scoring system based on the true positive and false positive detection probabilities for each scanner. We propose to substitute TP, TN, FP, and FN with true positive rate (TPR), true negative rate (TNR), false positive rate (FPR) and false negative rate (FNR) in the accuracy formula (*ACC*) from Section 7.1 and use the resulting value as the individual score of scanner i which becomes

$$S_i = \frac{P^T{}_i + 1 - P^F{}_i}{2} \tag{8}$$

Here, S_i is the individual score of scanner i, $P^T{}_i$ is the true positive rate or probability of scanner i and $P^F{}_i$ is the false positive rate or probability of scanner i. Since both (TNR+ + FPR) and (TPR+ + FNR) are equal to 1, TNR becomes (1 – FPR) or (1- $P^F{}_i$) and (TPR + TNR + FPR + FNR) becomes 2. Based on this score, we can sort all the N scanners in a descending order. Then, to get M best scanners, we can select the top M scanners from the ordered set of N scanners.

7.3. The dependency approximation method

To identify the best subset of scanners, we can use the dependency approximation table calculated for the Dependency Approximation model in Section 5. There should be two dependency approximation tables, one for true positive probabilities and one for the false positive probabilities. Each entry of the tables has two members: a scanner subset and a probability value which is either a true positive probability or a false positive probability depending on the table. In this way, there is a one-to-one correspondence between the two tables. That means, for every entry of the true positive table, there is a corresponding entry in the false positive table. We developed the algorithm FindBestSubset (shown in Algorithm 4), where the input is the dependency approximation tables and a given size M. Then, for each entry in the tables where the scanner set (a subset of N scanners) is of size M, we use Eq. (8) (line 8 in Algorithm 4) to calculate the accuracy score. The subset with the maximum accuracy score is returned. Here are some explanations on the functions used in the algorithm. The CopyList function (lines 1 and 2) is used to return the portion of

Algorithm 4 FindBestSubset($L_{T(1-N)}$, $L_{F(1-N)}$, M).

```
1: $L_{T(M)} \leftarrow$ CopyList($L_{T(1-N)}$, $M$)
2: $L_{F(M)} \leftarrow$ CopyList($L_{F(1-N)}$, $M$)
3: $maxValue \leftarrow -1$
4: $bestSubset \leftarrow \{\}$
5: for $i$ in range(0, size($L_{T(M)}$)) do
6:     $CP^T \leftarrow$ GetValue($L_{T(M)}[i]$)
7:     $CP^F \leftarrow$ GetValue($L_{F(M)}[i]$)
8:     $accValue \leftarrow (CP^T + +1 - CP^F)/2$
9:     if $accValue > maxValue$ then
10:        $maxValue \leftarrow accValue$
11:        $bestSubset \leftarrow$ GetCombination($L_{T(M)}$, $i$)
12:    end if
13: end for
14: return $bestSubset$
```

the lists $L_{T(1\text{-}N)}$ or $L_{F(1\text{-}N)}$ where the size of the scanner set is exactly *M*. The GetValue function (lines 6 and 7) is used to return the combined probability value for the entry in index *i*. The GetCombination function (line 11) is used to return the scanner combination or set for index *i*.

8. Numerical simulation

To verify the accuracy of our models and to answer the questions mentioned in Section 3, we performed several numerical simulation experiments. We developed Python programs that can simulate the scanning of a set of samples by a set of anti-virus scanners. In this section, we describe the setup of these experiments and their results in detail.

8.1. Simulation of the models

We defined a hypothetical set of 1000 malicious and 1000 benign samples and 10 anti-virus scanners. We randomly decided whether a sample is detected as malicious or not by a particular anti-virus scanner. The true labels of the samples are used to calculate the true positive and false positive detection rates of each individual anti-virus scanner. Then, for the ground truth case, the true labels of the samples are used again to calculate the combined true positive rate (CP^T) and false positive rate (CP^F) for all the threshold values ranging from 1 to 10. We calculated the combined detection rates (CP^T and CP^F) for all the threshold values using our models as well. We calculated the pairwise combined detection probabilities by calculating the ratio of the number of samples detected as malicious or benign by the scanners in a pair and the total number of samples in the common sample set. For DAM, we used the pairwise detection probabilities for all possible pairs of scanners to construct the dependency approximation table. Then, we calculated the accuracy values both for the ground truth case and for our models based on three metrics of evaluation, as described in Section 7.1.

As mentioned earlier, we randomly decided whether a sample is detected as malicious or not by an anti-virus scanner. To create different test sets with different detection rates for the anti-virus scanners, we enforced different maximum values so that all the anti-virus scanners will have a detection rate that is below the maximum value for that test set. This means, for example, if the maximum value is 90, all the anti-virus scanners (10 in our experiments) will have a maximum detection rate of 0.9 or 90%. We varied the maximum value to create all the test sets spanning all possible detection rates. The range of maximum values for true positive rates was from 50 to 95 and the range of maximum values for false positive rates was from 5 to 50.

In this simulation experiment, we tried to test uniformly for all detection rates within the selected range for an equal number (1000) of samples for both malicious and benign types. Although we could easily make the sample set unbalanced by using a much higher number for the benign set, we observed that since the simulation was done using a simple pseudo random function which would maintain the same detection rates regardless of the size of the sample set, an effort to make it unbalanced is unnecessary and would not provide any additional insight.

8.1.1. Analysis: detection probability based modeling is good enough to achieve a high accuracy that is very close to the maximum

To better illustrate our simulation results, we show the graphs of one test case, where the true positive rate and the false positive rate was limited to 80% and 10% respectively. Fig. 5(a) shows the graphs of combined true positive rates generated from the ground truth case and the models for different threshold values ranging from 1 to 10. Similarly, Fig. 5(b) shows the graphs of combined false positive rates calculated from ground truth case and our models for different threshold values. Here, CPM and DAM exhibit the best results among all three proposed models and follow the actual true trend very closely.

Fig. 6 shows the comparison of true accuracy values when using the true optimal threshold (labeled in the graph as *True Maximum*) and when using the thresholds calculated from our models for the example test case using three different evaluation metrics. We have also included the Simple Majority (SM) method as a straightforward majority voting method to compare against our models. The Simple Majority (SM) method uses the simple rule of deciding a sample as malicious if at least 51% of the scanners vote

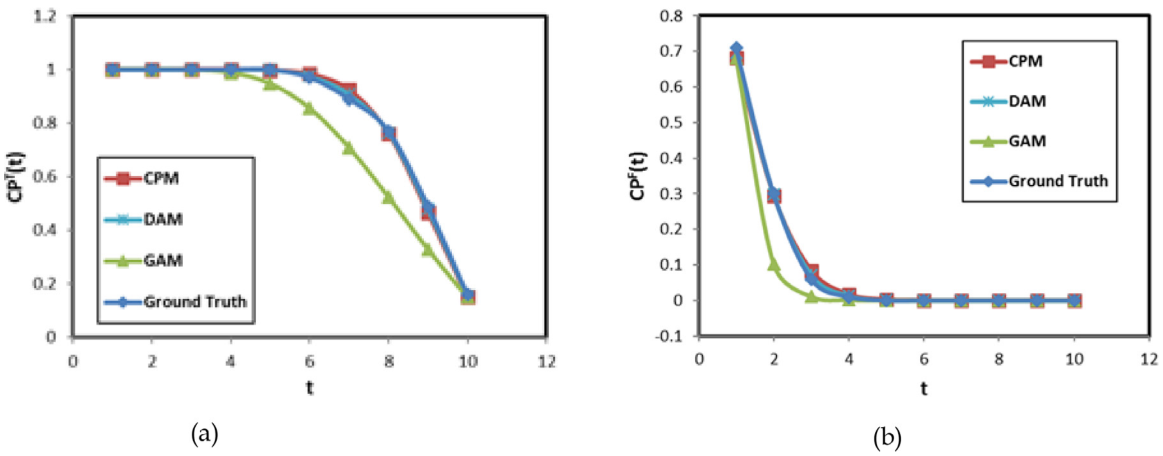


Fig. 5. Graphs of combined detection probabilities against different threshold values.

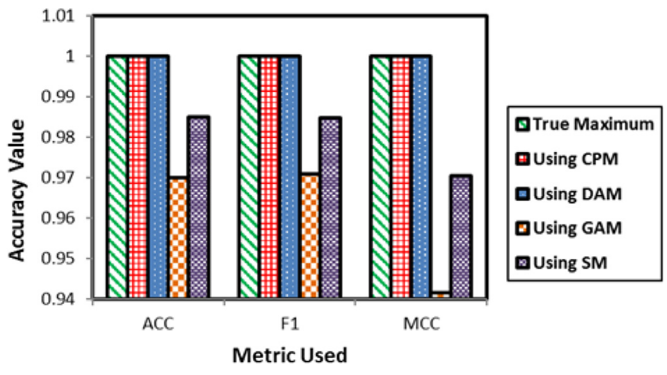


Fig. 6. Comparison of accuracy values using three evaluation metrics based on simulation results.

Table 2
Average deviation from maximum accuracy.

Metric Used	CPM	DAM	GAM	SM
ACC	0.03	0.02	0.1	0.15
F1	0.04	0.05	0.12	0.19
MCC	0.06	0.04	0.18	0.27

for it. In Fig. 6, we can see that both CPM and DAM perform as good as the true maximum, whereas GAM and SM are also very close to them. In this particular case, we see that SM slightly outperforms GAM.

To evaluate how our models perform against the ground truth cases, we varied the limiting maximum values for randomization and created different test cases. As mentioned earlier, the range of limiting maximum values for true positive rates was from 50 to 95 and the range for false positive rates was from 5 to 50. We varied the values with a step size of 5, creating a total of $10 \times 10 = 100$ test cases. Table 2 shows average deviation from the true maximum accuracy value for all three models and Simple Majority (SM) method based on three evaluation metrics we used. Results from Table 2 indicate that DAM performs best and is within 2% of true maximum accuracy, whereas CPM and GAM is within 3% and 10% respectively, based on the ACC metric. It is also evident from the results for SM that although it can outperform GAM in specific scenarios, on average it performs poorly compared to all our models.

The implication of these simulation results is that, our models, which are solely based on detection probabilities of the scanners, can be used to predict an optimal configuration which can achieve a high accuracy close to the maximum. This is very important and useful for the cases where we can only collect minimal information about the detection capabilities of the scanners.

8.2. Simulation of optimal size for scanner set (Q)

The optimal size of the scanner set (Q) refers to the minimum number of scanners in a scanner set that achieves the maximum accuracy value among all available N scanners. Here, the goal of our simulation test is to determine whether adding new scanners to a multi-scanner system can always improve or maintain the maximum accuracy. In other words, if we have a total of N scanners available, should we use all of them (i.e., $Q = N$), or is it possible to remove some scanners from the set (i.e., $Q < N$) to achieve maximum accuracy?

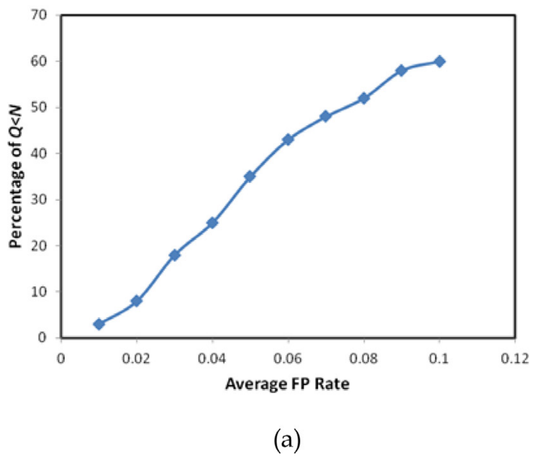
In the simulation test, we vary the average false positive detection rate of the scanners and calculate the value of Q. The value of N is selected as 10 as in previous tests. The value of average false positive rate is varied from 0.01 to 0.1 with a step size of 0.01. We run the tests for each average false positive rate value 100 times to get an average estimate.

8.2.1. Analysis: whether we should use all N scanners or not depends on the false positive rate

Fig. 7(a) shows the percentage of times when Q is less than N out of all instances as we increase the average false positive rate of scanners. Here, the increase is almost linear, and it reaches up to more than 50% when the average false positive rate reaches 0.1.

Fig. 7(b) shows the calculated average values of Q when N is 10, as we increase the average false positive rate. Here, the average value of Q almost linearly decreases with the increase in average false positive rate and it goes down by more than 10% when the false positive rate reaches 0.1.

Both graphs in Fig. 7 verify that if the false positive rates of the scanners are high enough, the optimal number of scanners that will yield the maximum accuracy can be lower than the total number of available scanners. This means, if the false positive rate is very low, it does not impact the overall accuracy. To explain, let's consider the extreme case where the false positive rates of all the scanners are zero. In this case, the accuracy of the multi-scanner system only depends on the true positive detection rates of the scanners. Therefore, adding a new scanner to the system where the new scanner also has a zero false positive rate will either improve or maintain the level of accuracy, since the new scanner can only either detect a previously undetected malicious sample, or detect an already detected sample. From this scenario, if we start increasing the false positive rates of the scanners, we must consider



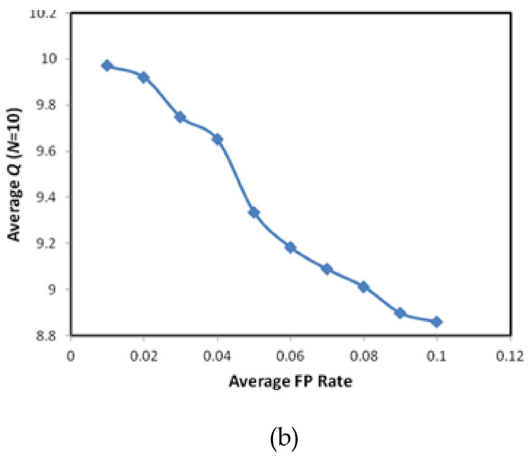


Fig. 7. Trends of changes in Q vs. average false positive rate.

the effect of a new scanner which also has a non-zero false positive rate. This means, the new scanner may falsely detect a benign sample as malicious, which will ultimately add to the overall false positive rate. The overall accuracy might still improve or stay the same if the new scanner's contribution to true positive is greater or equal to its contribution to false positive. However, as the false positive rate increases, the probability that the new scanner will add more to true positive than false positive decreases. Again, we can verify this from the fact that for a new scanner in a multi-scanner system that already has a high number of scanners, the probability of detecting an undetected malicious sample is lower than the probability of falsely detecting an undetected benign sample when the false positive rate is high enough. The graphs in Fig. 7 also confirm this trend. In conclusion, we can say that the decision of whether to use all available scanners in a total of N scanners or whether to go beyond that by adding a newly available scanner should be made based on the false positive detection rates of the scanners. In other words, we need to determine the optimal number of scanners (Q) to achieve maximum accuracy. Our models can be effectively used for this purpose.

8.3. Simulation of finding best subset of scanners

In Section 7.2, we proposed two methods for finding the best subset of scanners with size M. We conducted simulations to test how the performance of the subsets generated by these methods fit into the range of maximum accuracy values achieved by any M scanner subset.

Fig. 8 shows the graph for a sample simulation test done to compare the maximum accuracy values achieved by the best combination, the worst combination, and the combinations derived by the ranking method and the dependency approximation method. The individual scanner's true positive and false positive detection rates were randomized as in the previous simulation tests and were limited to a maximum value. In this test case, true positive rates were limited to 80% and false positive rates were limited to 5%. Here, our methods perform much better than the worst combination selected and perform almost at the same level as the best combination for higher M values. The dependency approximation method-based combinations perform slightly better than the ranking method based combinations for lower M values, which means if the dependency information of the scanners is available, we can make use of it to find better subset of scanners. We executed similar simulation test 100 times to get an average estimate of how our methods perform. This simulation experiment yielded the following results.

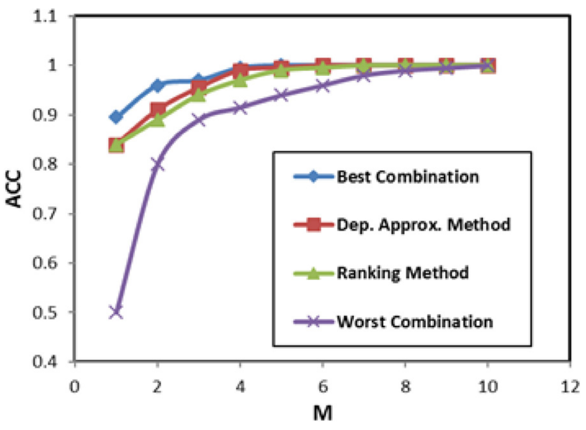


Fig. 8. Comparison of maximum accuracy by best and worst combinations and combinations generated by our methods based on simulation results.

8.3.1. Analysis: selecting the best scanners is crucial to achieve maximum accuracy

We saw that it is important to determine the optimal number of scanners to achieve maximum accuracy. But after the optimal number is determined, how important is it to choose which scanners to use? From the experimental results, we found that on average the difference between the best combination and the worst combination in terms of accuracy is 0.085 or 8.5%. This is a considerable gap, which means it is very important to optimize the selection of scanners in the subset.

8.3.2. Analysis: both ranking method and dependency approximation method can achieve an accuracy within 2% of the maximum

On average, our ranking method and dependency approximation method provides a combination which achieves an accuracy value that is 0.0195 and 0.013 lower than the maximum accuracy achieved by the best combination respectively and is 0.0655 and 0.072 higher than the maximum achieved by the worst combination respectively. This means that using these methods we can achieve at least 2% within the maximum and improve by more than 6.5% from the minimum. These results are calculated using the first metric (ACC).

8.3.3. Analysis: the dependency information of the scanners is useful to determine better subset of scanners

On average, the dependency approximation method based subset combinations can achieve slightly higher (0.0065) accuracy than the ranking method based subset combinations. This is due to our use of the pairwise dependency information to generate the dependency approximation table. Although this is a small improvement, we can argue that the more dependency information that is available to be used, the better subset combination can be identified.

9. Experimental evaluation using real data

9.1. Malware and goodware dataset

We discussed the practical issues associated with accurate computation of detection rates of the scanners in a real multi-scanner system in Section 9.5. For our experimental evaluation, we collected a large dataset of malware samples from VirusSign [25], which commercially provides a significant amount of high-quality malware samples. Our malware dataset consisted 38,789 malware samples in total. Our goodware dataset consisted of 21,624 benign portable executable (PE) binary files collected from SourceForge [26]. We downloaded these files by crawling the SourceForge website in order of user ratings to ensure they are not malicious. Table 3 lists the details of each of the malware and goodware datasets.

We divided both the malware and goodware dataset further into training and test sets. The training datasets are used to calculate individual true and false detection probabilities (P^T and P^F) for each anti-virus scanner. These values are used by our models to calculate combined detection probabilities ($CP^T(t)$ and $CP^F(t)$) according to our CPM formula (Eq. (7)) and DAM and GAM algorithms. Then, the test datasets are used to calculate the true combined detection probabilities ($CP^T(t)$ and $CP^F(t)$) for each threshold t. Table 4 lists the division of malware and goodware dataset into corresponding training sets and test sets. We used multiple test sets of varying sizes by dividing the full test set to add diversity into the experiments.

9.2. Experimental setup

We used online multi-scanning service VirusTotal for our experiments. VirusTotal generates scanning reports based on scanning

Table 3
Malware and goodware dataset.

Name	Source	Number of Samples	Period of Collection
Malware Dataset	VirusSign	38,789	April 26 to April 29, 2014
Goodware Dataset	SourceForge	21,624	July 20 to July 31, 2015

Table 4
Training and test sets.

Name	Number of Samples
Malware Training Set	28,789
Malware Test Set	10,000
Goodware Training Set	11,624
Goodware Test Set	10,000

Table 6
Distribution of test sets for combined accuracy test.

Test Set	Number of Malware Samples	Number of Goodware Samples
1	4000	4000
2	4000	2000
3	2000	4000

Table 5
List of anti-virus scanners.

Kaspersky	ESET-NOD32	VBA32
Antivir	GData	VIPRE
Agnitum	Ikarus	TrendMicro-HouseCall
Avast	K7GW	BitDefender
AVG	McAfee-GW-Edition	Emsisoft
Comodo	Malwarebytes	NANO-Antivirus
DrWeb	Sophos	Panda

performed by more than 70 anti-virus scanners (at the time of the writing). But not all the reports contain the same anti-virus scanners all the time. Therefore, we identified a set of 21 anti-virus scanners (listed in Table 5) that were common to all the generated scanning reports.

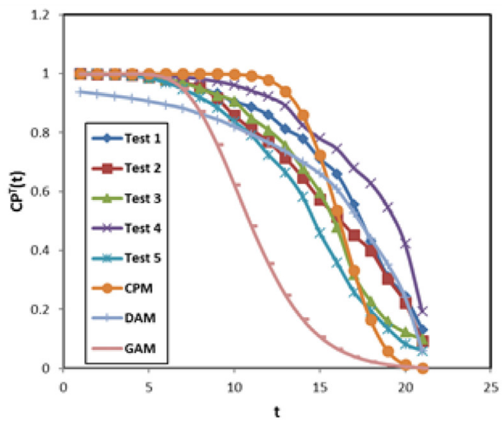
We developed a program in C#.NET that is based on the VirusTotal API to generate the scanning reports from VirusTotal, and another program in Python to parse and calculate our desired combined detection probability and accuracy values from them. We also implemented our models using Python.

9.3. Results and analysis

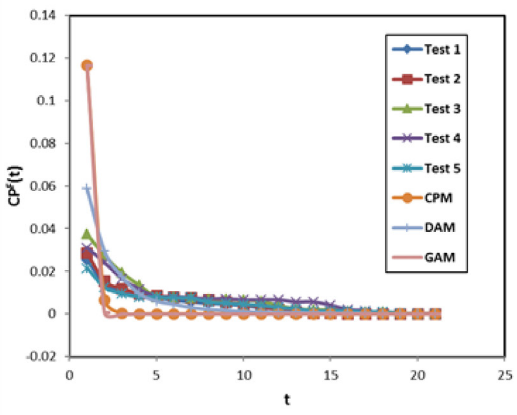
9.3.1. Analysis: detection probability based modeling can achieve an accuracy that is almost as good as the maximum

Fig. 9(a) shows the graphs of combined true positive detection probability ($CP^T(t)$) against threshold values (t) from 1 to 21. Here, we divide the full test set (both malware and goodware) into 5 test sets (*Test 1* to *Test 5* in the figure) containing 2000 samples each, which serve as our ground truth cases here. From the graphs, we can see that the combined true positive detection rate varies from *Test 1* to *Test 5*. In the graphs generated from the models, CPM and GAM generate a smooth mathematical curve, whereas DAM displays a trend which resembles the curves generated from true data, although we see some divergence from the other curves at the beginning. This is because we are using more information from actual or true data (pairwise dependency information) to eventually calculate the final values. A similar trend can be found in Fig. 9(b), which demonstrates the graphs of combined false positive detection probabilities ($CP^F(t)$) against threshold values (t) from 1 to 21.

We use the combined true and false positive detection probabilities to calculate accuracy values according to three evaluation metrics from Section 7.1 and use them to determine the optimal threshold. To add diversity in test sizes, we created 3 test sets from the malware and goodware test set according to Table 6. However, as we can see from the results, it does not have any effect on the evaluation. Fig. 10 shows the comparative graphs for these accuracy values. The accuracy values calculated using the models are the true accuracy values for the model-predicted optimal thresholds. Fig. 10(a) presents the comparative accuracy values for test set #1 (from Table 6) based on three accuracy metrics. Fig. 10(b) shows the comparison for all three test cases based on the first accuracy metric ACC. To compare against alternative straightforward methods, we have included here two additional methods, namely the Simple Majority (SM) method, where at least 51% vote from the scanners are needed to label a sample malicious, and the Fixed



(a)



(b)

Fig. 9. Comparison of graphs of combined detection probabilities against threshold values generated from ground truth test cases and our models.

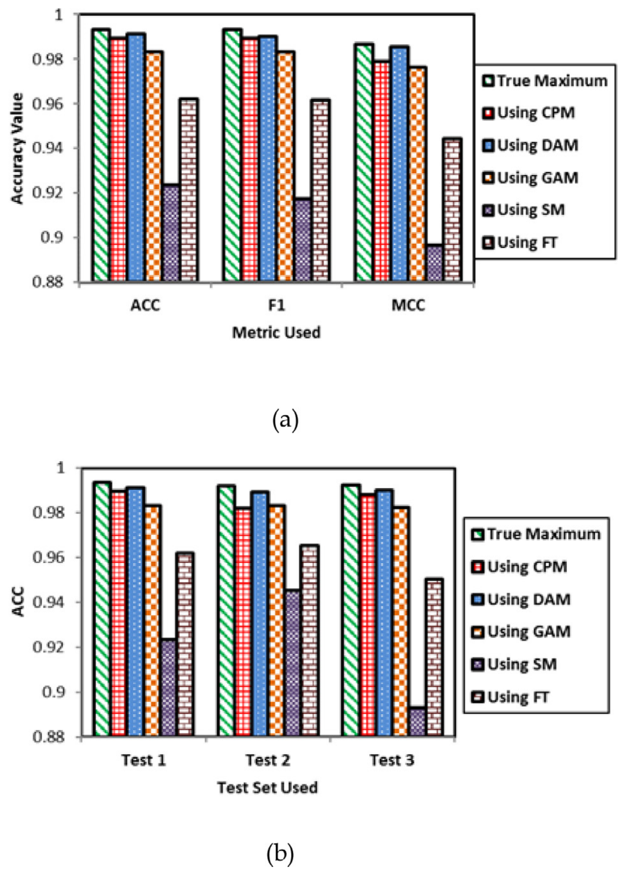


Fig. 10. (a) Comparison of accuracy values using three evaluation metrics based on real world test set 1, (b) Comparison of accuracy values using only ACC metric based on all three test sets.

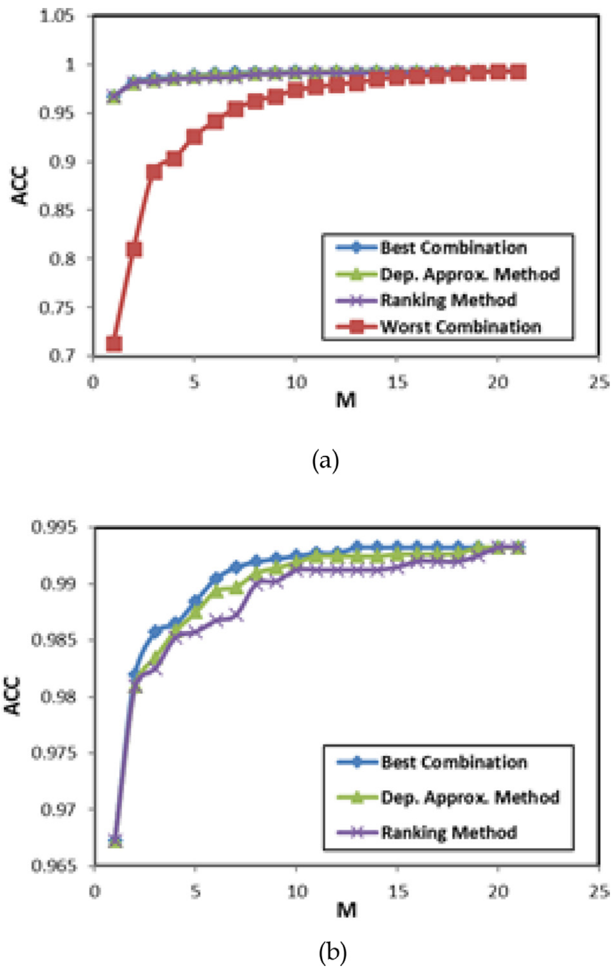


Fig. 11. (a) Comparison of maximum accuracy by best and worst combinations and combinations generated by our methods based on real world dataset, (b) Comparison of only the maximum accuracy by best combinations and combinations generated by our methods (scale changed to show finer details).

Threshold (FT) method, where a fixed threshold of 7 (1/3 of total number of scanners) is used. We can see that for all the test cases, the model-predicted accuracy values are very close (within 1%) to true maximum accuracy values. We also see that there is a very small difference in accuracy values among CPM, DAM, and GAM, where CPM and DAM perform better in comparison to GAM. Moreover, we see that both SM and FT methods perform poorly compared to our models.

Confirming our findings in Section 8.1, these results show that modeling based on detection probabilities can achieve an even better accuracy level in real scenarios. This also confirms the importance of considering detection capabilities when building a multi-scanner system.

9.3.2. Analysis: both ranking method and dependency approximation method can achieve a combination that is almost as good as the best combination

Next, we conduct all combination tests where we take a subset of M scanners from all N scanners and calculate maximum accuracy values for the best combination, the worst combination and the combinations from ranking method and dependency approximation method. Fig. 11 shows the graphs for this experiment done only on the test set #1 from Table 6. Experiments on test sets #2 and #3 yield similar results and so are omitted here for space constraints. In Fig. 11, we see that both the ranking method and dependency approximation method yield accuracy values that are very close to the maximum accuracy values achieved by the best combination and much higher than the maximum accuracy achieved by the worst combination. We have also calculated an average among all 3 test sets to find out the average difference of the accuracy values for the combinations. We found that on average the maximum accuracy value calculated using the ranking method and dependency approximation method is lower than the maximum accuracy for the best combination by 0.00164 and 0.00084 respectively and is higher than the maximum accuracy for the worst combination by 0.05468 and 0.04187 respectively. This means that our methods can provide a combination that is almost as good as the best combination (within at least 0.2% in terms of accuracy). Again, these results confirm our findings from Section 8.3 and show that in the real scenarios our ranking method and dependency approximation method performs even better. In addition, it also confirms that adding the dependency information helps in finding a better subset combination compared to straightforward ranking of the scanners.

9.3.3. Analysis: we should use all N *scanners to achieve maximum accuracy due to very low false positive rates in practical scenarios*

Another important observation from Fig. 11 is that the accuracy values tend to always increase with the increase of M, which means the value of Q is always equal to N. This is because the average false positive rate for all the scanners is 0.00864 which is lower than 0.01. This also verifies our simulation results from Section 7.2, where we have seen that for very low average false positive rates Q is almost equal to N and the probability of Q being lower than N is very low. Therefore, in practical scenarios we can say that more scanners will never reduce accuracy of a multi-scanner system and we should use all N scanners to achieve a maximum accuracy. There can be only two cases where this might not be true. First, if a subset of all N scanners can already achieve a maximum possible accuracy of 1, and second, there is a cost issue where using all N scanners will be too expensive, since as we increase the number of scanners, the cost continues to increase but the additional improvement in accuracy becomes extremely low.

In Section 8.2, we explained why the decision to use all N scanners depend on the false positive rates of the scanners. The same reasoning is applicable here. If we assume an average false positive rate of less than 0.01 as zero false positive rate, then the trend of increasing accuracy as we increase the number of scanners can be understood very easily. The implications from these results can be further extended to provide guidelines for adding a new scanner beyond the total of N scanners as follows:

1 Whether the new scanner should be added to the multi-scanner system depends on the false positive detection probability associated with the new scanner. Using our models, we can effectively calculate whether the $(N + 1)$-scanner system (with the new scanner included) has higher accuracy than the previous N-scanner system and make a decision on the new scanner.

2 If the decision is not to increase the number of scanners to $N + 1$, then we can check whether we should replace one of the existing scanners in the N-scanner system with the new scanner. A straightforward approach is to use our ranking method and rank the new scanner among the existing set of scanners. A more extensive approach would be to replace each of the existing scanners by the new scanner one by one and determine whether the overall accuracy is improved or not.

9.4. Runtime analysis and comparison of the models

A comparison of the models in terms of runtime analysis is given in Table 7, which shows that CPM and DAM are more computationally expensive than GAM. The reason behind this is the combinatorial component in the formula for CPM and DAM. For DAM, there is also a one-time execution cost of $O(N^2\binom{N}{N/2})$ to calculate the Dependency Approximation table values, which is not included here (it takes several hours to calculate, but it is a one-time cost that won't be needed in future scans). We have calculated the actual execution time from our experiments for each model, which is also listed in Table 7. The execution time was measured on an Intel Core i3 2.10 GHz laptop for a scenario where N was assigned to be 20. We found that CPM takes almost more than 6 min and DAM takes slightly less than a minute to execute, whereas GAM takes about 1 ms. Based on the runtime complexity given in Table 7 and the actual execution time that we measured, a proportional calculation done for the same computing platform and 30 scanners ($N = 30$) shows that it will take more than 200 h for CPM and more than 17 h for DAM to finish execution. This result shows that both CPM and DAM are not the best choice in terms of scalability, and GAM provide an efficient alternative. If we want to reduce the execution time even more, we can consider using a subset of M scanners instead of all N scanners, where $M < N$. If it is desired to make the best tradeoff between scalability and accuracy, GAM could be preferable to either CPM or DAM. On the other hand, in any practical scenario where accuracy is highly prioritized than scalability, CPM and DAM should be the option to use. For example, if we are creating a ground truth dataset, where the slightest inaccuracy could have a major impact, we should go for CPM and DAM instead of GAM.

Table 7
Comparison of the models.

Criteria	CPM	DAM	GAM
Runtime Complexity	$O(N^2\binom{N}{N/2})$	$O(2^N)$	$O(NlgN)$
Actual Execution Time	402.55 s	57.38 s	0.00099 s

9.5. Discussion on practical applications and challenges

In this section, we discussed some applicative scenarios and challenges for a real-world implementation of our proposed models and algorithms. Some applicative scenarios are discussed below:

1. Depending on whether we have the dependency information of the scanners, we have the option to use the following combinations of multi-scanner models and optimal scanner subset selection methods:
   a. If dependency information is not available:
      i. CPM and Ranking method,
      ii. GAM and Ranking method,
   b. If dependency information is available:
      i. DAM and Dependency Approximation method.
2. The optimal scanner subset computation (using either Ranking method or Dependency Approximation method) should only be necessary for the following cases:
   a. Initial selection of scanners for the multi-scanner system,
   b. Every time a scanner is added to or removed from the set of scanners,
   c. Every time there is a change to the detection probabilities of one or more scanners.
3. The Dependency Approximation table (for DAM and Dependency Approximation method) computation should only be necessary for the following cases:
   a. Initial computation for the multi-scanner system,
   b. Every time there is a change in the dependency information of one or more pairs of scanners.
4. It might seem that the need for computation of optimal subset and Dependency Approximation table is quite frequent, but we anticipate that this is only true for the very early stages of development of the multi-scanner system. As the detection and dependency information of the scanners become more stabilized, the need for such computation should become less frequent.

We acknowledge that there are several practical challenges in implementing our proposed solution, as discussed below:

1. One of the biggest practical challenges for our proposed system is to compute accurate detection rates for the scanners in the system. The accuracy of this detection rate computation mainly depends on the following two things:
   a. The accuracy of the labels of the samples in the dataset that is used for the computation. In other words, we need ground truth datasets for both malware and goodware.
   b. Considerable amount of detection results for each scanner, where the samples from the ground truth dataset are scanned by the scanners to produce these detection results.

   For the first requirement, we need to establish ground truth datasets. One way to establish a ground truth dataset is by manual vetting and analysis of the samples in the set, unless there already exist commercially available ground truth datasets with claims of very high (almost 100%) accuracy. But it is extremely difficult and time consuming to do that on a large scale. Our proposed multi-scanner system models can be used to generate a ground truth dataset, by initially using a manually vetted ground truth dataset as the training set and iteratively adding new labeled samples to that ground truth dataset.

For the second requirement, we need considerable amount of detection results for each scanner. One way to achieve this is by collecting past detection history of the scanners for several years. But it might be difficult to ensure consistency, validity and authenticity of the results, since anti-virus scanners are constantly updated and detection results for a given sample might differ from one version to another. Moreover, the detection history must be based on our established ground truth dataset and from a reliable source. Therefore, one solution to generate considerable amount of detection results is to follow the same iterative approach as we mentioned for developing a ground truth dataset. Both the development of a ground truth dataset and the accumulation of detection results can be done together by starting with a manually vetted ground truth dataset and generating the detection results for that dataset using the actual scanners that the multi-scanner system should own. Then, computing detection rates for the scanners and using our models we can identify more ground truth labels and again use them to generate more detection results. Here, we should mention that we have used the VirusSign malware database and highly rated binary files from SourceForge as the ground truth dataset and VirusTotal as the source for detection results in this paper, which is only for experimental purposes and obviously less than ideal for a real scenario. An important aspect in this process of developing the database of ground truth samples and detection results is the optimal selection of scanners. Since initially we are starting with no detection rates, we must include all available scanners to compute their detection rates. But as we proceed further into the process with each iteration, we can use the newly computed detection rates to decide on an optimal selection of scanners, if necessary. We acknowledge that it might take several such iterations before we can have a considerable database of ground truth samples and detection results to effectively utilize our multi-scanner system.

2. Another significant challenge is to maintain the consistency and accuracy of the detection results with update and upgrade of the anti-virus scanners. Since a previously undetected sample could now be detected with the newest version of a scanner, we must re-scan previously scanned samples by the newest version to maintain consistency and accuracy of detection results and detection rates of the scanner. As the ground truth datasets grow, it might become a time-consuming task to re-scan all the samples for every update of the scanners.
3. We acknowledge that it is extremely difficult to completely remove bias from anti-virus detection results. However, for practical purposes and for our models to work with reasonable accuracy, we can select a dataset with a reasonable non-bias through the following steps:
   a. Identify datasets with as much diversity as possible. Diversity should be considered with respect to the following:
      i. Origin
      ii. First occurrence
      iii. Behavior
   b. Make the dataset as large as possible.
4. The size of the dataset to be considered unbiased depends on the number of anti-virus scanners that will be used in a multi-scanner detection system and the level of diversity that can be ensured in that dataset. The more anti-virus scanners are used, the larger the dataset it should be. Also, if we can ensure a high level of diversity in a relatively smaller dataset, it could still be considered unbiased.
5. For better comparative analysis and collection of dependency information among the scanners, we recommend that the scanners are tested on the same dataset. But we also acknowledge that it might be infeasible to ensure that in a practical scenario.

10. Conclusion and future work

Protecting computer systems from malicious contents has been and will remain a top priority for cyber security practitioners, and the first and foremost step in the protection mechanism is the detection of malware and other malicious contents. In this paper, we provided a new set of guidelines in achieving the optimal detection capabilities of malware using multiple anti-virus scanners. To capture the behavior of a multi-scanning malware detection system, we presented two theoretical approximation models based on only the individual detection capabilities or ratings of the member scanners in the system and one additional model based on pairwise dependency information among the scanners in addition to the individual detection rates. These models can effectively help us in finding the optimal threshold to achieve maximum accuracy in an N-scanner system, which we verified with experimental evaluations. Furthermore, we learn the following important lessons through these models:

1. Whether we should use all available scanners or not depend on the false positive detection rates of the scanners. Addition of new scanners with poor false positive detection rates can lower the overall accuracy rather than improving it.
2. Selection of scanners is crucial to achieve the maximum accuracy.
3. Ranking the scanners based on accuracy scores helps in selecting a combination of scanners that is almost as good as the best combination.
4. Incorporating pairwise dependency information of the scanners into the selection of best scanners is beneficial.

These findings along with our models together provide a set of important guidelines for any multi-scanner detection system consisting of only third-party anti-virus scanners where very little information is available about the internal design and working mechanism of them.

In the future, we plan to extend our models to incorporate additional information such as internal mechanism and behavior of the anti-virus scanners and include experimental evaluation for such cases. In addition, we anticipate to further extend this work into other areas of malicious content detection, such as intrusion detection and anti-spam filtering.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2019.107027.

References

[1] J.A. Morales, S. Xu, R. Sandhu, Analyzing malware detection efficiency with multiple anti-malware programs, in: *Proc. ASE/IEEE International Conference on BioMedical Computing*, December, 2012.
[2] M. Cukier, I. Gashi, B. Sobesto, V. Stankovic, Does malware detection improve with diverse antivirus products? An empirical study,, in: Proc. 32nd International Conference on Computer Safety, Reliability and Security (SAFECOMP), 2013.
[3] J. Oberheide, E. Cooke, F. Jahanian, CloudAV: N-version antivirus in the Network Cloud, in: Proc. 17 th USENIX Security Symposium, 2008.
[4] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. Clausen, O. Kiraz, K.A. Yuksel, S.A. Camtepe, S. Albayrak, Static analysis of executables for collaborative malware detection on android, in: Proc. IEEE International Conference on Communications (ICC'09), 2009.

[5] C.J. Fung, D.Y. Lam, R. Boutaba, A decision making model for collaborative malware detection networks, School of Computer Science *Technical Report: CS-2013-01*, University of Waterloo, Canada, 2013.
[6] C.J. Fung, D.Y. Lam, R. Boutaba, Revmatch: a decision model for collaborative malware detection, Department of Computer Science Technical Report CS-2013-01, University of Waterloo, 2013.
[7] C. Silva, P. Sousa, P. Verissimo, Rave: replicated antivirus engine, in: Proc. IEEE International Conference on Dependable Systems and Networks Workshops (DSN-W), 2010, pp. 170–175.
[8] M. Marchetti, M. Messori, M. Colajanni, Peer-to-peer architecture for collaborative intrusion and malware detection on a large scale, Inf. Secur. (2009) 475–490.
[9] M. Colajanni, D. Gozzi, M. Marchetti, Collaborative architecture for malware detection and analysis, in: Proc. the 23rd International Information Security Conference, The International Federation for Information Processing (IFIP), 278, 2008, pp. 79–93. Volume.
[10] H. Lu, X. Wang, J. Su, SCMA: scalable and collaborative malware analysis using system call sequences, Int. J. Grid Distrib. Comput. 6 (2) (2013) 11–28.
[11] VirusTotal, available at http://www.virustotal.com. (accessed in December 2015).
[12] Jotti, available at http://virusscan.jotti.org. (accessed in December 2015)
[13] VirSCAN, available at http://www.virscan.org. (accessed in December 2015)
[14] File2Scan, available at http://www.file2scan.net. (accessed in December 2015)
[15] Metadefender, available at https://www.metadefender.com. (accessed in December 2015)
[16] HerdProtect, available at http://www.herdprotect.com. (accessed in December 2015)
[17] HitmanPro, available at http://www.surfright.nl/en/hitmanpro. (accessed in December 2015)
[18] SecureAPlus, available at http://www.secureaplus.com/Main/index.php. (accessed in December 2015)
[19] Multi-AV, available at http://multi-av.thespykiller.co.uk. (accessed in December 2015)
[20] Emsisoft, available at http://www.emsisoft.com. (accessed in December 2015)
[21] G Data, available at https://www.gdata-software.com. (accessed in December 2015)
[22] Accuracy, available at https://en.wikipedia.org/wiki/Accuracy_and_precision. (accessed in June 2016)
[23] F1 Score, available at https://en.wikipedia.org/wiki/F1_score. (accessed in June 2016)
[24] Matthews Correlation Coefficient, available at https://en.wikipedia.org/wiki/Matthews_correlation_coefficient. (accessed in June 2016)
[25] VirusSign, available at http://www.virussign.com. (accessed in April 2014)
[26] SourceForge, available at http://www.sourceforge.net. (accessed in July 2015)
[27] A. Kantchelian, et al., Better malware ground truth: techniques for weighting anti-virus vendor labels, in: Proc. the 8th ACM Workshop on Artificial Intelligence and Security, 2015.
[28] M. Chiriac, Tales from cloud nine, in: Virus Bulletin Conference, Geneva, Switzerland, 2009, pp. 1–46. http://www.virusbtn.com/pdf/conferenceislides/2009/Chiriac-VB2009.pdf. Sep. 24available at.
[29] A. Algaith, I. Gashi, B. Sobesto, M. Cukier, S. Haxhijaha, G. Bajrami, Comparing detection capabilities of antivirus products: an empirical study with different versions of products from the same vendors, in: Proc. 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop, 2016.
[30] A. Aldini, F. Martinelli, A. Saracino, D. Sgandurra, Detection of repackaged mobile applications through a collaborative approach, Concurrency and Computation: Practice and Experience, John Wiley and Sons Ltd., 2015.
[31] L. Song, H. Huang, W. Zhou, W. Wu, Y. Zhang, Learning from big malwares, in: *Proc. the 7th ACM SIGOPS Asia-Pacific Workshop on Systems,*August, 2016.
[32] M. Sebastián, R. Rivera, P. Kotzias, J. Caballero, AVclass: a tool for massive malware labeling, in: Proc. International Symposium on Research in Attacks, Intrusions, and Defenses. Springer International Publishing (RAID), 2016.
[33] R.M. Valdovinos, J.S. Sanchez, Combining Multiple Classifiers With Dynamic Weighted Voting, HAIS, 2009.
[34] T. Górecki, M. Krzyśko, Regression methods for combining multiple classifiers, Commun. Stat. (2015).
[35] A. Mohaisen, O. Alrawi, Av-meter: an evaluation of antivirus scans and labels, in: Proc. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Cham, Springer, 2014.
[36] B. Miller, et al., Reviewer integration and performance measurement for malware detection, in: Proc. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Cham, Springer, 2016.
[37] J. Charlton, et al., Measuring relative accuracy of malware detectors in the absence of ground truth, in: Proc. IEEE Military Communications Conference (MILCOM), IEEE, 2018.
[38] M. Hurier, et al., On the lack of consensus in anti-virus decisions: metrics and insights on building ground truths of android malware, in: Proc. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Cham, Springer, 2016.
[39] K. Rieck, et al., Automatic analysis of malware behavior using machine learning, J. Comput. Secur. 19 (4) (2011).
[40] R. Perdisci, W. Lee, N. Feamster, Behavioral clustering of HTTP-based malware and signature generation using malicious network traces, NSDI. 10 (2010) Vol..
[41] L. Chen, C. Yagemann, and E. Downing, "To believe or not to believe: validating explanation fidelity for dynamic malware analysis," *arXiv preprint arXiv:1905.00122* (2019).
[42] M.Z. Rafique, J. Caballero, Firma: malware clustering and network signature generation with mixed network behaviors, in: Proc. International Workshop on Recent Advances in Intrusion Detection, Springer, 2013.
[43] K. Rieck, et al., Learning and classification of malware behavior, in: Proc. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2008.
[44] R. Perdisci, A. Lanzi, W. Lee, McBoost: boosting scalability in malware collection and analysis using statistical classification of executables, in: Proc. IEEE Annual Computer Security Applications Conference (ACSAC), 2008.
[45] J. Stiborek, T. Pevný, M. Rehák, Multiple instance learning for malware classification, Expert Syst. Appl. 93 (2018).
[46] U. Bayer, et al., Scalable, behavior-based malware clustering, NDSS 9 (2009) Vol..
[47] W. Huang, J.W. Stokes, MtNet: a multi-task neural network for dynamic malware classification, in: Proc. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, 2016.
[48] R. Vinayakumar, et al., Robust intelligent malware detection using deep learning, IEEE Access 7 (2019).
[49] A. Pektaş, T. Acarman, Classification of malware families based on runtime behaviors, J. Inf. Secur. Appl. 37 (2017).
[50] A. Atzeni, et al., Countering android malware: a scalable semi-supervised approach for family-signature generation, IEEE Access 6 (2018).
[51] A. Damodaran, et al., A comparison of static, dynamic, and hybrid analysis for malware detection, J. Comput. Virol. Hacking Tech. 13 (1) (2017) 1–12.
[52] W. Wong, M. Stamp, Hunting for metamorphic engines, J. Comput. Virol. 2 (3) (2006) 211–229.
[53] S. Attaluri, S. McGhee, M. Stamp, Profile hidden Markov models and metamorphic virus detection, J. Comput. Virol. 5 (2) (2009) 151–169.
[54] T. Singh, "Support Vector Machines and metamorphic malware detection," Master's report, Department of Computer Science, San Jose State University (2015), http://scholarworks.sjsu.edu/etd_projects/409/.
[55] S. Deshpande, Y. Park, M. Stamp, Eigenvalue analysis for metamorphic detection, J. Comput. Virol. Hacking Tech. 10 (1) (2014) 53–65.
[56] R.K. Jidigam, T.H. Austin, M. Stamp, Singular value decomposition and metamorphic detection, J. Comput. Virol. Hacking Tech. 11 (4) (2015) 203–216.
[57] C. Annachhatre, T.H. Austin, M. Stamp, Hidden Markov models for malware classification, J. Comput. Virol. Hacking Tech. 11 (2) (2015) 59–73.
[58] Y.H. Choi, et al., Toward Extracting Malware Features For Classification Using Static and Dynamic Analysis, in: Proc. 8th International Conference on Computing and Networking Technology (INC, ICCIS and ICMIC), IEEE, 2012, pp. 126–129.
[59] M. Eskandari, Z. Khorshidpour, S. Hashemi, HDM-Analyser: a hybrid analysis approach based on data mining techniques for malware detection, J. Comput. Virol. Hacking Tech. 9 (2) (2013) 77–93.
[60] T. Singh, et al., Support vector machines and malware detection, J. Comput. Virol. Hacking Tech. 12 (4) (2016) 203–212.
[61] B. Zhang, et al., Malicious codes detection based on ensemble learning, International Conference on Autonomic and Trusted Computing, Springer, Berlin, Heidelberg, 2007.
[62] Y.-B. Lu, et al., Using multi-feature and classifier ensembles to improve malware detection, J. CCIT 39 (2) (2010) 57–72.
[63] E. Menahem, et al., Improving malware detection by applying multi-inducer ensemble, Comput. Stat. Data Anal. 53 (4) (2009) 1483–1494.

Muhammmad N. Sakib received his BS degree in computer science and engineering from Bangladesh University of Engineering and Technology, Bangladesh in 2008 and his PhD degree from the Department of Computer Science and Engineering, University of South Carolina, USA. From 2008 to 2010, he worked for Commlink Infotech Ltd. as a member of the research and development team. From 2010 to 2012, he worked as a research assistant with the Department of CSE, University of South Carolina. From 2012 to 2015 he worked as a teaching assistant for the same department. From 2015 he has been working for Itron, Inc. He received SPARC graduate research grant from University of South Carolina in 2014. His research interests include network security and privacy.

Chin-Tser Huang received the BS degree in computer science and information engineering from the National Taiwan University, Taipei, Taiwan in 1993, and the MS and PhD degrees in computer sciences from the University of Texas at Austin in 1998 and 2003, respectively. He joined the faculty at the University of South Carolina in Columbia in 2003 and is now a professor in the Department of Computer Science and Engineering. His research interests include network security, network protocol design and verification, cloud computing, and distributed systems. He is the director of the Secure Protocol Implementation and Development (SPID) Laboratory at the University of South Carolina. He is the author (along with Mohamed Gouda) of the book Hop Integrity in the Internet, published by Springer in 2005. He is a senior member of IEEE and a senior member of ACM.

Ying-Dar Lin is a Distinguished Professor of computer science at National Chiao Tung University (NCTU), Taiwan. He received his Ph.D. in computer science from the UCLA in 1993. He was a visiting scholar at Cisco Systems in San Jose (2007–2008) and the CEO at Telecom Technology Center, Taipei (2010–2011). Since 2002, he has been the founder and director of Network Benchmarking Lab (NBL), which reviews network products with real traffic and has been an approved test lab of the Open Networking Foundation (ONF) since July 2014. He also cofounded L7 Networks Inc. in 2002, which was later acquired by D-Link Corp. His research interests include network security, wireless communications, and network cloudification. His work on multi-hop cellular was the first along this line, and has been cited over 750 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014–2017), and ONF Research Associate. He serves on the editorial boards of several IEEE journals and magazines, and is the Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST). He published a textbook, Computer Networks: An Open Source Approach, with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).