future internet

MDPI

Article

Optimal Weighted Voting-Based Collaborated Malware Detection for Zero-Day Malware: A Case Study on VirusTotal and MalwareBazaar

Naonobu Okazaki [1,†], Shotaro Usuzaki [1,*,†], Tsubasa Waki [1], Hyoga Kawagoe [1], Mirang Park [2], Hisaaki Yamaba [1] and Kentaro Aburada [1]

[1] Faculty of Engineering, University of Miyazaki, 1-1 Gakuen-Kibanadai-Nishi, Miyazaki-shi 889-2192, Miyazaki, Japan; oka@cs.miyazaki-u.ac.jp (N.O.); hm20017@student.miyazaki-u.ac.jp (H.K.); yamaba@cs.miyazaki-u.ac.jp (H.Y.); aburada@cs.miyazaki-u.ac.jp (K.A.)
[2] Faculty of Information Technology, Kanagawa Institute of Technology, 1030 Shimo-Ogino, Atsugi-shi 243-0292, Kanagawa, Japan; mirang@nw.kanagawa-it.ac.jp
* Correspondence: usuzaki@cs.miyazaki-u.ac.jp
† These authors contributed equally to this work.

Abstract: We propose a detection system incorporating a weighted voting mechanism that reflects the vote's reliability based on the accuracy of each detector's examination, which overcomes the problem of cooperative detection. Collaborative malware detection is an effective strategy against zero-day attacks compared to one using only a single detector because the strategy might pick up attacks that a single detector overlooked. However, cooperative detection is still ineffective if most anti-virus engines lack sufficient intelligence to detect zero-day malware. Most collaborative methods rely on majority voting, which prioritizes the quantity of votes rather than the quality of those votes. Therefore, our study investigated the zero-day malware detection accuracy of the collaborative system that optimally rates their weight of votes based on their malware categories of expertise of each anti-virus engine. We implemented the prototype system with the VirusTotal API and evaluated the system using real malware registered in MalwareBazaar. To evaluate the effectiveness of zero-day malware detection, we measured recall using the inspection results on the same day the malware was registered in the MalwareBazaar repository. Through experiments, we confirmed that the proposed system can suppress the false negatives of uniformly weighted voting and improve detection accuracy against new types of malware.

Keywords: malware detection; collaborative security; VirusTotal; MalwareBazaar

Citation: Okazaki, N.; Usuzaki, S.; Waki, T.; Kawagoe, H.; Park, M.; Yamaba, H.; Aburada, K. Optimal Weighted Voting-Based Collaborated Malware Detection for Zero-Day Malware: A Case Study on VirusTotal and MalwareBazaar. *Future Internet* **2024**, *16*, 259. https://doi.org/10.3390/fi16080259

Academic Editors: Weizhi Meng and Christian D. Jensen

Received: 4 July 2024
Revised: 19 July 2024
Accepted: 20 July 2024
Published: 23 July 2024

1. Introduction

As the Internet becomes a critical piece of infrastructure for our lives, malware threats are increasing yearly. Malware steals confidential information, destroys important data, and sometimes uses the data as collateral for blackmail, money laundering, and other highly malicious crimes. In recent years, a large-scale distributed denial-of-service attack was carried out by exploiting IoT devices infected with a malware program called Mirai [1]. In 2017, a large number of computers around the world were infected with ransomware called WannaCry [2]. Furthermore, in 2022, a targeted attack by a malware application called Emotet was attached in phishing emails [3].

Although accurate and quick malware detection is essential, a single detection method can overlook the attack. The number of new malware attacks is constantly being observed each year according to the report [4]. This report indicates that the malware detection method should collect intelligence about new malware in real-time to respond to its activity. Many malware detection systems have been widely studied, but they can still overlook the malware depending on the type. Even today, as the accuracy of machine learning

technology continues to improve, this method still has problems in terms of usability and safety.

While collaborative malware detection has been widely studied to tackle the issue of hard-to-detect attacks [5–8], they are still insufficient to detect new malware. These methods conduct voting based on the detection results obtained from multiple anti-virus vendors testing the same sample and use the majority result to determine the sample's maliciousness. While studies [5–8] showed that this approach could detect malware more accurately than the results of a single anti-virus inspection, the majority decision would negatively impact the accuracy of the judgment. For example, even if reliable detection engines with high accuracy determine that a suspicious file is malware, the malware will be overlooked when most classifiers determine that the file is benign. Based on this concern, these majority-vote-based methods might miss the new type of malware because most anti-virus engines lack sufficient information to detect the new malware at that time.

To consider the reliability of the anti-virus engines in a final decision, we weighted the voting value of malware detection by anti-virus engines according to their accuracy history. This method can be expected to improve the detection accuracy by reducing false negatives, that is, overlooked malware. This study mainly examined the accuracy of collaborative malware detection when weighting is set to the optimal setting based on VirusTotal results. To verify the actual detection accuracy against unknown malware, we used a malware repository service called MalwareBazaar. By inspecting malware registered in MalwareBazaar on the same day as it was registered, we simulatively measured the detection accuracy of unknown malware with insufficient information.

The remainder of this paper is structured as follows. Section 2 describes the background for the research domain of malware detection. Section 3 provides the key concept, architecture, and detailed components of the proposed system. Section 4 reports the evaluation experiment for the proposed system, and Section 5 gives and discusses its results. Section 6 and Section 7 discuss the differences and originality of our work and the limitations on interpreting the detection results. In Section 8, we summarize this paper.

The main contributions of this paper are as follows:

- An investigation of the actual detection accuracy of each anti-virus engine on VirusTotal [9] using actual malware samples. Combining multiple anti-virus engines could improve detection accuracy based on the observation that each engine has different strengths when it comes to the various malware categories.
- A demonstration of the potential of an optimal weighted-voting-based malware detection system to improve detection accuracy against unknown malware. We evaluated the overall recall when the weights for each anti-virus engine were assigned according to the above findings. We inspected the malware the same day it was first registered in the MalwareBazaar repository. This overall recall shows that false negatives for weighted voting decrease compared to the results of uniform weighting, even if the type of malware is new.
- A verification of whether the procedures for malware detection experiments conducted in this study enabled the evaluation of the detection accuracy against unknown malware. We examined the same malware files under the same conditions after more than 20 days to confirm that detection accuracy improved. As a result, we clarified that the evaluation of the detection accuracy for unknown malware can be simulated according to the above procedures.

2. Background

The signature-based detection method detects malicious programs by registering hashes or binaries of malware files as signatures in advance and matching them with samples at the time of inspection [10]. Therefore, the signature detection method can detect already known malware but fails to detect unknown malware. Because a delay occurs between the time when an anti-virus vendor discovers malware and the time when the

vendor registers the signature, computers are at risk of being infected with the newly discovered malware during this delay.

Heuristic detection is a method to determine whether a file is a malware application by extracting and registering behavioral features such as API call sequences and machine language opcodes from known malware in advance and comparing the program code from the file analysis to see if there is anything that behaves similarly to the registered features. Data mining and machine learning techniques are used to register the behavioral features of malware [11,12]. This detection method can deal with subspecies and unknown malware that are difficult to detect by the signature-based method. Therefore, it cannot detect new types of malware that behave in an uncommon manner. Although the accuracy of detection is improving with the development of machine learning technology, it is still not realistic to detect malware using heuristic detection methods alone, and most heuristic detection methods are used to compensate for the shortcomings of signature-based detection methods.

The behavior detection method, also known as dynamic heuristic detection, involves executing the sample code in an isolated environment, such as a virtual sandbox or other isolated environments, and monitoring its behavior and changes in the computer's internal environment to detect malicious entities [13,14]. However, malware developers attempt to circumvent behavior detection by incorporating features into their programs that hide the malware's malicious behavior. In addition, false positives and negatives are expected to occur due to the ambiguity of the decision criteria, and the method is also time-consuming and burdensome for the computers that deploy it. As with the heuristic detection method, this method is mostly used to compensate for the shortcomings of the signature detection method.

Collaborative security [5–8,15–17] is an approach in which multiple security systems and organizations share information and make security-related decisions. The advantages of this approach include rapid response to threats and optimization of each organization's resources by sharing information among multiple organizations and systems. In ref. [5], the author proposed a blockchain-based collaborative malware detection system that enables instantaneous and tamper-resistant detection by multiple anti-virus vendors. The anti-virus vendors join a consortium blockchain network and provide the detection results obtained by their own anti-virus engine. These results are used to calculate the maliciousness to determine whether a sample file is malware based on majority-voting-style malware detection. Throughout the experiment, it was shown that majority voting using detection results from multiple anti-virus engines was effective in reducing the number of false positives. However, depending on the voting results at the time of malware identification, detection results output from multiple anti-virus engines may be ignored even though they are accurate, resulting in unacceptable detection omissions. George et al. [7] tackled the issue of the risk of malicious Android Package Kit files causing malicious behavior on mobile devices that use the Android platform. As a solution, the authors proposed a system that classifies the functions or behaviors of mobile malware and performs malware detection by prediction and majority voting using a machine learning model on a blockchain node. Experimental results showed that the proposed system has better detection accuracy than a single malware detector.

In recent research, ensemble-based attack detection methods that combine multiple machine- and deep-learning-based detectors have been proposed [18–21]. These studies use multiple detectors to overcome the weaknesses of the machine and deep learning, such as a limited amount of training data, an imbalance between normal and abnormal samples, and the selection of optimal hyperparameter settings. In the simple strategy, the results are ensembled based on majority voting [18] and veto voting [18,19]. The veto voting determines the "malware" regardless of the number of votes, even if one vote claims the file is malware. The literature [20,21] makes the final malware detection based on the predicted class probabilities from multiple detectors. To ensemble the results from multiple detectors, Islam et al. [20] attempted weighted voting predictions based on the prediction accuracy

of each detector. Xue and Zhu [21] used a soft voting strategy to ensemble the multiple detection results. The soft voting strategy is one of the majority voting strategies that averages the class probabilities from multiple detectors and obtains the answer according to the maximum average class probability. This strategy can be useful in an environment where each detector can give the class probabilities of all categories. Although these papers mentioned weighted majority voting, they do not clarify how weights are calculated.

3. Proposed Method

3.1. Aim of This Research

Our research focuses on how to reflect the reliability of each anti-virus engine in the value of their votes on voting-based collaborative malware detection. Most voting-based malware detection systems [5–8] are limited by not considering each anti-virus engine's detection performance, and they use votes with uniform weights to determine malware. The potential drawback is the possibility of overlooking malware when the system determines a malware sample as benign based on the votes of multiple-classifiers with low detection accuracy, even if the results of detection engines with high detection accuracy judge it as malicious. According to the previous studies, there will be a bias in the categories that each engine is more likely to detect, depending on which signatures the signature-based method has or which features the behavior method uses for analysis.

3.2. Architecture

As shown in Figure 1, the proposed method comprises one or more control servers and multiple anti-virus engines. The control server mainly inspects the malware acquired from users through multiple anti-virus engines and makes the final decision based on those results. The anti-virus engines are assumed to present as Web APIs or applications bundled together if the vendor allows it. The system workflow follows the five steps illustrated in Figure 1: (1) When a user uploads a file to the proposed service, the control server receives it and (2) asks the anti-virus engines to inspect it via an API. (3) Each detection result from each anti-virus vendor is aggregated onto the control server, and (4) then the server makes a final decision according to the algorithm detailed in Section 3.3.3. Finally, (5) the system returns the output to the users.

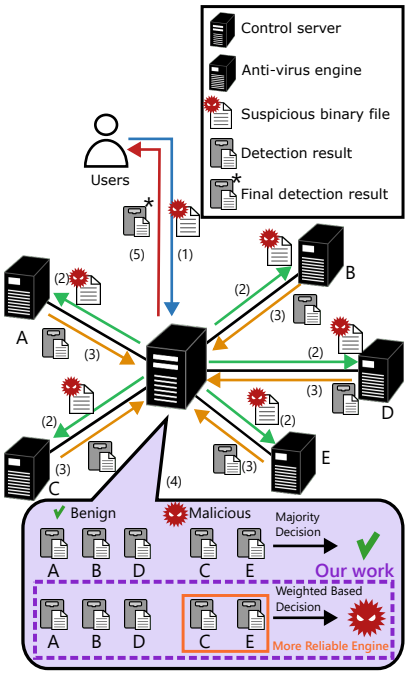

Figure 1. Architecture of the proposed system. See Section 3.2 for an explanation of the numbers in parentheses.

3.3. Malware Detection Algorithm

The proposed system detects malware according to the following steps:

1. Express the detection results from multiple anti-virus engines (detailed in Section 3.3.1).
2. Assign an appropriate weight for the vote from each anti-virus engine (detailed in Section 3.3.2).
3. Calculate the maliciousness to make a final decision on malware detection (detailed in Section 3.3.3).

Table 1 shows the notation used in Sections 3.3.1–3.3.3.

Table 1. Notation used in Section 3.

Notation	Description
C	Total number of malware categories handled
K	Total number of active anti-virus engines
M	Anomaly score calculated by the proposed system
T	Threshold for the anomaly score
V_{b}	Total score of benign judgment votes
V_{m}	Total score of malicious judgment votes
W	Weight matrix ($W \in \mathbb{R}^{C \times K}$)
$w_{\mathrm{high}}, w_{\mathrm{mid}}, w_{\mathrm{low}}$	Weights of vote, $w_{\mathrm{high}} > w_{\mathrm{mid}} > w_{\mathrm{low}}$
$D_{\mathrm{high}}, D_{\mathrm{low}}$	Criterion for assigning vote weights for the detection rate
$S_{\mathrm{high}}, S_{\mathrm{low}}$	Criterion for assigning vote weights for the anomaly score
R_{ck}	Detection rate for category c, which is from the k-th anti-virus engine ($R_{\mathrm{ck}} \in [0, 1]$)
$\hat{s}$	Normalized anomaly score s from the anti-virus engine

3.3.1. Expression of Detection Results

The results of each detection should be expressed numerically and uniformly to enable the system to handle different result representations from multiple anti-virus engines. Each engine might output different anomaly scores: signature-based methods might present a discrete value of 0 or 1, while machine learning methods might express the maliciousness as a probability value between 0 and 100. Some anti-virus engines might also present additional information, such as malware categories.

In this study, the output of every engine was normalized to a value between 0 and 1 to obtain a voting value. The proposed system expresses benign results as "0" and malicious results as "1". If the value output by the engine does not fall within the range, it is normalized using the maximum value the engine can output.

3.3.2. Assignment of Appropriate Weights

In this study, we introduced a method of weighting the detection results output from anti-virus engines by applying an appropriate weight value according to the analysis history. The proposed system sets different priorities at multiple levels depending on the contents of the detection results output by each anti-virus engine.

To reflect the reliability of detection results, the proposed system uses a weight matrix $W \in \mathbb{R}^{C \times K}$, where C is the total number of malware categories handled and K is the total number of active anti-virus engines. The malicious voting weight for category c of the k-th anti-virus engine W_{ck} takes three weighting levels $w_{\mathrm{high}} > w_{\mathrm{mid}} > w_{\mathrm{low}}$. If an anti-virus engine provides a binary-category classification rather than a multi-category classification, the system assigns the same weight to all categories.

The three weighting levels are assigned by Equation (1) based on the detection rate R_{ck}. R_{ck} is a value that indicates the performance of anti-virus engines, which is calculated

in advance by attempting the detection process against a set of binary files whose true categories are already known.

$$W_{\mathrm{ck}} = \begin{cases} w_{\mathrm{low}} & \text{if } 0 \leq R_{\mathrm{ck}} < D_{\mathrm{low}}, \\ w_{\mathrm{mid}} & \text{if } D_{\mathrm{low}} \leq R_{\mathrm{ck}} < D_{\mathrm{high}}, \\ w_{\mathrm{high}} & \text{if } D_{\mathrm{high}} \leq R_{\mathrm{ck}} \leq 1. \end{cases} \tag{1}$$

The metrics that can be used as R_{ck} are not limited to any specific one if they are related to detection performance. The system administrator might adopt the recall to reduce false negatives, the F-value to balance precision and recall, or some other metric.

Unfortunately, this rule alone does not allow for the calculation of weights when the reported malware category is not included in the set of categories maintained by the system. In this case, the system takes the following measures to make the results as useful as possible. Some anti-virus engines using machine learning techniques might calculate the anomaly score s. If the anomaly score is included in the anti-virus engine response, W_{ck} is assigned a weight value by Equation (2) where $\hat{s}$ is normalized s to the range of $[0, 1]$. If there is no anomaly score response, W_{ck} is assigned w_{low}.

$$W_{\mathrm{ck}} = \begin{cases} w_{\mathrm{low}} & \text{if } 0 \leq \hat{s} < S_{\mathrm{low}}, \\ w_{\mathrm{mid}} & \text{if } S_{\mathrm{low}} \leq \hat{s} < S_{\mathrm{high}}, \\ w_{\mathrm{high}} & \text{if } S_{\mathrm{high}} \leq \hat{s} \leq 1. \end{cases} \tag{2}$$

3.3.3. Calculation of the Anomaly Score for the Final Decision

The calculation of the anomaly score is inspired by the methods used in [5]. A sample file is determined as malware based on whether the calculated anomaly score $M (0 \leq M \leq 1)$ is greater than or equal to a predefined threshold T. The anomaly score is calculated regardless of the malware category to avoid missing attacks even when category classification accuracy is unstable. Although this score is the same as that in Fuji et al. [5], the difference is that weights are used to add up V_{m} as shown in Algorithm 1, which includes all three steps.

$$M = V_{\mathrm{m}} / (V_{\mathrm{m}} + V_{\mathrm{b}}) \tag{3}$$

Algorithm 1 Proposed malware detection algorithm

function DETECTION($file$)
  $V_{\mathrm{m}} = 0, V_{\mathrm{b}} = 0$
  for all $k \leftarrow K$ **do**
    if k-th anti-virus engine determines *file* is malicious **then**
      $w = w_{\mathrm{low}}$
      if Determined category c is in the set of maintained categories **then**
        $w \leftarrow W_{ck}$
      else
        if k-th anti-virus engine response includes the anomaly score s **then**
          $\hat{s} \leftarrow$ normalize(s)
          $w \leftarrow$ get_weight($\hat{s}$) ▷ Assign the weight value by Equation (2)
        end if
      end if
      $V_{\mathrm{m}} \leftarrow V_{\mathrm{m}} + 1 \times w$ ▷ Apply the weight for a malicious judgment vote
    else
      $V_{\mathrm{b}} \leftarrow V_{\mathrm{b}} + 1$
    end if
  end for
  $M = V_{\mathrm{m}} / (V_{\mathrm{m}} + V_{\mathrm{b}})$ ▷ Calculate the anomaly score
  if $M \geq T$ **then**
    return The final decision is "malicious"
  else
    return The final decision is "benign"
  end if
end function

4. Experiment

4.1. Experiment 1: Area-of-Expertise of Each Anti-Virus Engine

To rank anti-virus engines in terms of detection results, we first investigated whether malware categories exist for which anti-virus engines have a detection advantage. It is necessary to determine the priority of weighting, which is indispensable for selecting the appropriate size of weights according to the record of an anti-virus engine's detection results. We collected malware files and those hashes for which malware categories have already been identified using the online malware repository service MalwareBazaar [22]. MalwareBazaar provides malware hashes and files for a target malware category by specifying the name as the search tag (Figure 2). We used this search tag as the correct label for malware categories.

In this evaluation experiment, we used the malware analysis web service VirusTotal [9] as multiple anti-virus engines to assess the same sample files and to acquire the results. VirusTotal provides online malware detection using more than 70 anti-virus engines. Users can inspect files and websites and search for inspection results using URLs and file hashes. We obtained the analysis results in the JSON format using the VirusTotal API. The prototypes were created as Python scripts and used in the experiments.

We then used the results to calculate and compare the average recall for each anti-virus engine's malware category, hoping to clarify the distribution of detection rates and the existence of each engine's area of expertise. The recall is calculated by $tp/(tp + fn)$, where tp is true positive and fn is false negative.

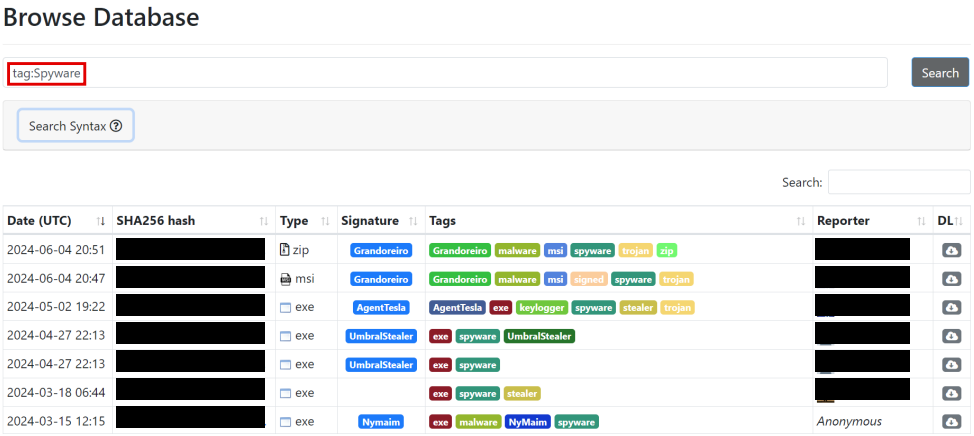

Figure 2. Screenshot of how malware samples were retrieved in MalwareBazaar for experiment 1. Users can search for malware tagged with XX by entering tag:XX. This figure shows the search result for "tag:Spyware" as shown in the red box at the top.

4.2. Experiment 2: Zero-Day Malware Detection Performance

This experiment aimed to verify that weighted voting-based malware detection could improve the recall compared to non-weighted voting-based methods.

In our experiments, we collected malware samples using MalwareBazaar to verify the accuracy of both methods for detecting unknown malware. From a practical standpoint, evaluating detection performances for including unknown malware files is desirable because new malware files are created daily. However, it is difficult to intentionally collect the latest unknown malware immediately after its appearance. Because new malware files are uploaded to the MalwareBazaar repository daily, the acquired malware would likely be the latest, including unknown malware.

We retrieved the malware samples registered in MalwareBazaar and inspected them on VirusTotal the same day (see Table 2). Each group had up to 250 samples because the VirusTotal API is limited to calling 500 samples per day and consumes the resources of two APIs to upload a file and download the corresponding detection result. Groups D and E had fewer samples than the others, as the number of registered malware samples on that day did not exceed 250. The detection result was evaluated by recall as described in Section 4.1.

Table 2. Malware samples used in experiment 2 (collected in 2023).

Group	A	B	C	D	E	F	G
Collection and Inspection Date	10/30	10/31	11/01	11/05	11/06	11/07	11/08
Number of Samples	250	250	250	98	249	250	250

We compared the recall between the proposed system and Fuji et al. [5] based on the uniformly weighted voting method. Hereafter, Fuji et al.'s method is referred to as the **conventional method**. The threshold value T was set to 0.5, as in the environment conditions in [5]. This allowed us to compare the recall between the proposed system with weighted voting and the conventional method with simple majority voting. The voting weights for each anti-virus engine were calculated using the recall scores observed in experiment 1. We empirically set the weights as $w_{\text{high}} = 3$, $w_{\text{mid}} = 2$, $w_{\text{low}} = 1$ and $D_{\text{low}} = S_{\text{low}} = 0.65$, $D_{\text{high}} = S_{\text{high}} = 0.85$.

4.3. Experiment 3: Accuracy over Time

In this experiment, we verified whether the malware samples in experiment 2 likely contained unknown malware by seeing if they became detected over time. As mentioned in Section 4.2, measuring accuracy against truly unknown malware is challenging. The results of experiment 2 alone cannot determine whether it was truly unknown to those anti-virus engines.

Therefore, we conducted detection experiments on the same malware again after a period of time had passed. We then measured and compared the recall scores of experiment 2's results to verify whether detection accuracy had improved. In general, malware that has just been observed would be overlooked if it is a new or variant type. Still, as time passes and more information is gathered, detecting attacks of new malware should gradually become possible. We re-inspected the same samples as those used in experiment 2 more than 20 days after the first inspection to compare the change in accuracy over time. Thus, the samples used in the experiment were all the same as those in the results of experiment 2 (Table 2). Each sample was re-inspected on the date shown in Table 3.

Table 3. Malware samples used in experiment 3. The groups were the same as those used in experiment 2.

Group	A	B	C	D	E	F	G
Re-inspection Date	11/21	11/22	11/23	11/27	11/28	11/29	11/30

5. Results and Discussion

5.1. Experiment 1

This experiment provided a sufficient basis for using areas-of-expertise survey results as criteria for weighting votes. Table A1 in the Appendix A shows the average malware recall for each anti-virus engine, calculated based on the detection results by VirusTotal for a total of six malware categories, including Spyware, Backdoor, Trojan, Ransomware, Adware, and Worm. This result shows that even when the same malware was tested, differences in recall scores were caused by differences in anti-virus engines, indicating that each anti-virus engine has areas-of-expertise depending on the malware category.

5.2. Experiment 2

The experimental results showed that the proposed system was superior to the conventional method in terms of recall when 7 days of samples were used as the detection target. Figure 3 shows the result of recall between the conventional method and the proposed method for each sample group. Our method improved recall by about 0.1 across all samples, indicating that weighted voting can reduce false negatives.

Therefore, the proposed system detects malware more accurately than the uniformly weighted voting and is also effective for detecting unknown malware that has not yet been registered.

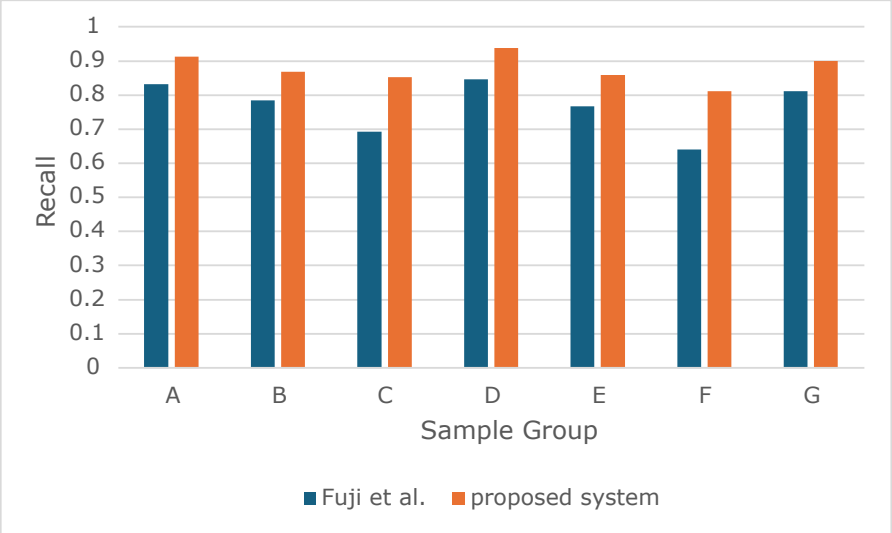

Figure 3. Recall scores of the proposed system and conventional method [5].

5.3. Experiment 3

As a result of the experiment, the results of experiment 2 can be interpreted as the results when using unknown malware for some anti-virus engines. Figure 4 shows the recall results between the first inspection date and the re-inspection date for each sample group. The experimental results showed that the malware recall scores improved with time.

One factor contributing to this result was that the malware pattern databases of some of the anti-virus engines used for detection via VirusTotal were updated within 20 days. Thus, this result indicates that the malware used in experiment 2 was undetectable in the early stages after the malware's appearance.

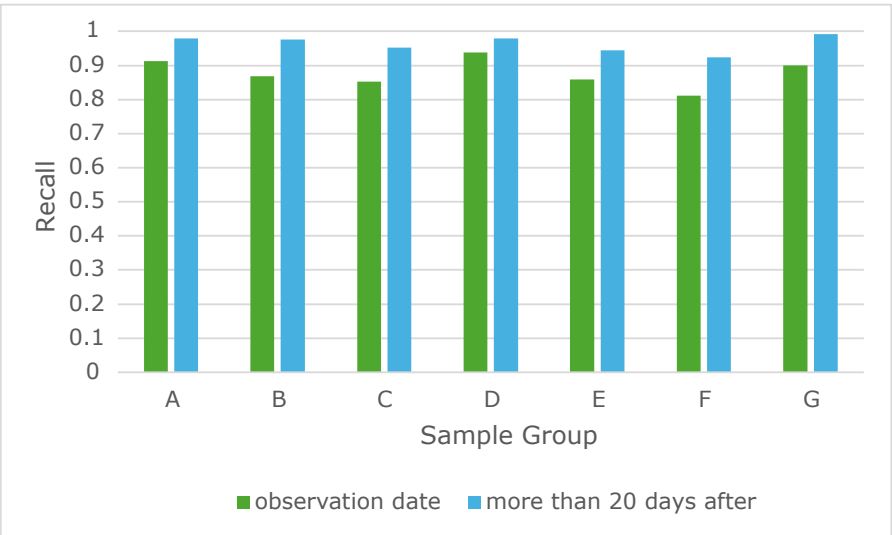

Figure 4. Recall scores of the proposed system over time. The results shown in "observation" are the same as the "proposed system" values in Figure 3.

6. Closely Related Research

Davies et al. [6] proposed a ransomware detection approach based on a cumulative malicious score through various benign-or-malicious binary classification tests to determine

the final outcome. Similar concepts can also be seen in the literature [7,23]. These studies did not consider the weights of each analysis, whereas this study provided insight that detection accuracy can be improved by setting appropriate weights.

Many studies [24–26] used the VirusTotal results to automatically generate correct labels for the supervised learning method. These studies mainly investigated whether VirusTotal detection results could provide correct labels for more accurate machine-learning training. They inspected malware samples obtained from public datasets [24], manually created datasets [25,26], or the *distribute* API on VirusTotal [25]. The *distribute* API enables us to retrieve the latest files uploaded to VirusTotal from users worldwide. We evaluated the detection accuracy against malware recently registered in MalwareBazaar, and used the tags as correct labels. This indicates that our results are not based solely on VirusTotal judgments, but may approximate more general results of the data labeled by annotators. However, it is important to interpret our results cautiously, as the labels may not always be tagged by reputable MalwareBazaar users.

Fung et al. [23] conducted research most closely related to our study. Their approach determines maliciousness by using the classification history as feedback from multiple detectors based on binary-classification. Unlike their approach, our work investigated the detection performance of multiple malware categories.

Similar to our work, Cocca et al. [27] evaluated the accuracy of anti-virus engines listed on VirusTotal on a large dataset collected by MalwareBazaar. They were interested in the prospect that there could be different detection results between the anti-virus engines. However, this research was not necessarily conducted on the same day it was registered; thus, unlike our research, it did not investigate the detection performance of new types of malware.

7. Future Work and Limitations

Since some notes apply to interpreting our findings, we discuss these notes by showing this study's future work and limitations.

The proposed system's evaluation experiments were conducted under the condition that all samples were malware. The false positive rate must still be evaluated by applying the system to benign files to demonstrate detection performance accurately. Generally, collecting unbiased benign files to evaluate false positives fairly is challenging. Appropriate methods should be considered, such as using manually created files as a ground truth dataset for evaluation, while allowing for some bias, as Zhu et al. [25] did. If our system is implemented by applying the VirusTotal API, each anti-virus engine's false positives are shown on VirusTotal statistics based on user reports [28]. These statistics could be useful in deciding which anti-virus engine to use in practice.

Furthermore, the detection performance reported in this study must be interpreted with caution as to whether it truly represents the accuracy of detecting unknown malware. Although the MalwareBazaar submission policy states, "Please refrain from uploading malware samples older than 10 days to MalwareBazaar", the freshness and correctness of the samples depend on whether the users that upload adhere to the standards. Therefore, it is necessary to re-examine the best way to collect samples for experiments in the future.

Although this study used VirusTotal, some studies have pointed out that some scanners alter their verdict even if the same file is inspected [24,25]. Hence, only scanners known to be stable and suitable scanners should be incorporated into the detection. For instance, Salem et al. [24] removed unstable scanners by a certainty score calculated by tallying up the output labels analyzing the same application multiple times. This approach also helps prevent adversarial scanners that consistently return irresponsible results from trying to mislead the final decision in cooperative systems, even without VirusTotal.

8. Conclusions

Highly accurate malware detection methods are needed to cope with the ever-increasing number of malicious attacks causing damage to society. Although malware detection tech-

niques such as signature-based, heuristic-based, and behavior-based methods have been widely used, different detection methods are suitable for different attacks. Given the many new types and variants that emerge, a single technique is not enough to detect all malware.

While collaborative malware detection could improve detection performance by integrating multiple detection results, these approaches can even overlook new types of malware. Most methods output the final result by majority vote, which means they miss attacks that are difficult to detect for many detection methods. Since variants and unknown malware are developed by attackers to evade detection, most anti-virus engines do not have sufficient evidence to detect malware when it appears quickly.

This study verified the detection accuracy of collaborative malware detection using VirusTotal and MalwareBazaar to achieve high accuracy even against unknown malware. We developed a prototype system that applies weights to the malware detection results of anti-virus engines based on their areas-of-expertise. Even if the malware is new, false negatives can be reduced by prioritizing dynamic analysis methods with high detection accuracy.

We evaluated the detection accuracy against unknown malware to confirm that the optimization of weights determined based on reliability could deal with new types of malware. By inspecting malware that appeared in MalwareBazaar on the same day, we measured the detection accuracy of unknown malware with insufficient information.

Through the evaluation, we confirmed the new system's superiority over uniformly weighted voting in terms of detection accuracy improvement due to the suppression of false negatives. We also verified that the new system can improve detection accuracy over time by examining the transition of malware recall.

Author Contributions: Conceptualization, T.W.; formal analysis, S.U. and T.W.; funding acquisition, N.O., M.P., H.Y. and K.A.; investigation, T.W.; methodology, T.W.; software, T.W.; supervision, N.O.; validation, S.U.; visualization, S.U.; writing—original draft, N.O., S.U., T.W. and H.K.; writing—review and editing, M.P., H.Y. and K.A. All authors will be informed about each step of manuscript processing including submission, revision, revision reminder, etc. via emails from our system or assigned Assistant Editor. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Japan Society for the Promotion of Science under KAKENHI grant numbers JP22K12013, JP23K11089, JP24K14917, JP24K14948.

Data Availability Statement: The original data presented in the study are openly available in https://drive.google.com/drive/folders/1vGcBdbjDusgzoMPWyVCRGx_vvM7w_g3s?usp=sharing (accessed on 19 July 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Average Recall Scores for All Anti-Virus Engines

Table A1. Average recall scores for anti-virus engines on VirusTotal. [1] "N/A" means that no detection results were obtained. Numbers in parentheses are the number of samples in each category.

Engine Name	Spyware (144)	Backdoor (122)	Trojan (500)	Ransomware (500)	Adware (500)	Worm (497)
Acronis	0.076	0.197	0.222	0.418	0.694	0.584
Ad-Aware	0.611	0.443	0	0.166	0.04	0.531
AhnLab-V3	0.889	0.697	0.926	0.886	0.366	0.799
Alibaba	0.861	0.541	0.898	0.838	0.926	0.797
ALYac	0.229	0.861	0.882	0.980	0.982	0.930

Table A1. *Cont.*

Engine Name	Spyware (144)	Backdoor (122)	Trojan (500)	Ransomware (500)	Adware (500)	Worm (497)
Antiy-AVL	0.500	0.730	0.868	0.978	0.980	0.742
APEX	0.194	0.434	0.884	0.704	1.00	0.821
Arcabit	0.681	0.803	0.880	0.996	0.994	0.797
Avast	0.493	0.893	0.934	0.920	0.982	0.946
Avast-Mobile	0.701	0.041	0.026	0.022	N/A[1]	0.010
AVG	0.493	0.893	0.934	0.920	0.982	0.954
Avira	0.944	0.893	0.842	0.750	0.980	0.867
Avware	N/A	N/A	N/A	N/A	N/A	0.002
Baidu	0.014	0.049	0.006	0.032	0	0.288
BitDefender	0.854	0.959	0.882	0.994	0.992	0.980
BitDefenderFalx	0.660	0.025	0.028	N/A	N/A	N/A
BitDefenderTheta	0.132	0.295	0.232	0.530	0.986	0.680
Bkav	0.090	0.213	0.358	0.422	0.100	0.592
CAT-QuickHeal	0.681	0.303	0.614	0.586	0.114	0.455
ClamAV	0.083	0.279	0.432	0.550	0.114	0.565
CMC	0	0	0	0	0	0.004
Comodo	0.243	0.393	N/A	0.100	0.086	0.425
CrowdStrike	0.194	0.484	0.860	0.580	0.486	0.795
Cybereason	0.118	0.262	0.352	0.514	0.006	0.638
Cylance	0.194	0.508	0.950	0.764	0.838	0.825
Cynet	0.910	0.811	0.944	0.952	0.980	0.907
Cyren	0.646	0.631	0.912	0.922	0.980	0.895
DeepInstinct	0.049	0.180	0.936	0.316	0.112	0.298
DrWeb	0.861	0.754	0.832	0.918	0.748	0.851
eGambit	0.014	0.025	N/A	N/A	N/A	0.042
Elastic	0.222	0.590	0.910	0.824	0.986	0.873
Emsisoft	0.826	0.943	0.872	0.976	0.972	0.952
Endgame	N/A	N/A	N/A	N/A	N/A	0.002
ESET-NOD32	0.951	0.893	0.964	0.888	0.988	0.964
F-Prot	0	N/A	N/A	N/A	N/A	0.002
F-Secure	0.389	0.377	0.834	0.492	0.426	0.495
FireEye	0.861	0.975	0.908	0.994	0.994	0.990
Fortinet	0.924	0.820	0.966	0.914	1.00	0.897
GData	0.861	0.967	0.964	0.990	0.990	0.980
Google	0.535	0.615	0.926	0.930	0.980	0.626
Gridinsoft	0.132	0.295	0.886	0.554	0.358	0.547
Ikarus	0.931	0.779	0.962	0.932	0.964	0.873
Invincea	N/A	N/A	N/A	N/A	N/A	0.034
Jiangmin	0.153	0.582	0.308	0.718	0.070	0.419

Table A1. *Cont.*

Engine Name	Spyware (144)	Backdoor (122)	Trojan (500)	Ransomware (500)	Adware (500)	Worm (497)
K7anti-virus	0.208	0.541	0.876	0.856	0.008	0.799
K7GW	0.889	0.574	0.910	0.864	0.008	0.799
Kaspersky	0.924	0.869	0.958	0.922	0.988	0.972
Kingsoft	0.569	0.320	0	0.340	0.498	0.392
Lionic	0.882	0.836	0.916	0.906	0.922	0.813
Malwarebytes	0.229	0.443	0.934	0.762	0.866	0.783
MAX	0.799	0.951	0.882	0.984	0.994	0.966
MaxSecure	0.208	0.525	0.682	0.726	0.770	0.684
McAfee	0.924	0.721	0.968	0.926	0.986	0.875
McAfee-GW-Edition	0.861	0.877	0.978	0.958	0.962	0.922
Microsoft	0.917	0.877	0.940	0.966	0.990	0.944
MicroWorld-eScan	0.854	0.959	0.912	0.994	1.00	0.978
NANO-anti-virus	0.312	0.352	0.334	0.624	0.098	0.584
nProtect	N/A	N/A	N/A	N/A	N/A	0.002
Paloalto	0.146	0.393	0.712	0.352	0.502	0.531
Panda	0.215	0.541	0.882	0.876	0.068	0.746
Qihoo-360	0.035	0.033	N/A	N/A	N/A	0.093
Rising	0.167	0.713	0.916	0.948	0.540	0.897
Sangfor	0.194	0.672	0.904	0.810	0.626	0.879
SentinelOne	0.132	0.434	0.698	0.562	0.098	0.799
Sophos	0.750	0.689	0.970	0.918	0.982	0.869
SUPERAntiSpyware	0.028	0.025	0.220	0.132	0	0.113
Symantec	0.757	0.738	0.772	0.938	0.892	0.903
SymantecMobileInsight	0.625	0.033	0.032	N/A	N/A	N/A
TACHYON	0.035	0.082	0.084	0.276	0	0.308
tehtris	0.035	0.057	0.102	0.100	0	0.167
Tencent	0.840	0.762	0.912	0.924	0.674	0.877
TheHacker	N/A	N/A	N/A	N/A	N/A	0.002
TotalDefense	0	0	N/A	N/A	N/A	0.050
Trapmine	0.069	0.238	0.650	0.430	1.00	0.485
TrendMicro	0.208	0.721	0.794	0.932	0.848	0.779
TrendMicro-HouseCall	0.229	0.754	0.812	0.680	0.894	0.787
Trustlook	0.667	0.033	0.032	0	N/A	0
VBA32	0.208	0.508	0.644	0.688	0.620	0.811
VIPRE	0.653	0.770	0.950	0.990	0.988	0.851
VirIT	0.076	0.123	0.750	0.590	0.352	0.362

Table A1. *Cont.*

Engine Name	Spyware (144)	Backdoor (122)	Trojan (500)	Ransomware (500)	Adware (500)	Worm (497)
ViRobot	0.056	0.270	0.206	0.496	0.164	0.356
Webroot	0.139	0.369	0.668	0.472	0.124	0.364
WhiteArmor	N/A	N/A	N/A	N/A	N/A	0
Xcitium	0.160	0.385	0.652	0.628	0.518	0.280
Yandex	0.160	0.385	0.256	0.350	0.006	0.296
Zillya	0.292	0.508	0.264	0.748	0.002	0.531
ZoneAlarm	0.535	0.500	0.964	0.654	0.554	0.628
Zoner	0.035	0.016	0.040	0.048	0	0.147

References

1. The Strange Story of the Teens behind the Mirai Botnet. Available online: https://spectrum.ieee.org/mirai-botnet/ (accessed on 11 June 2024).
2. Savita, M.; Patil, M. A brief study of wannacry threat: Ransomware attack 2017. *Int. J. Adv. Res. Comput. Sci.* **2017**, *8*, 1938–1940.
3. Boyarchuk, O.; Mariani, S.; Ortolani, S.; Vigna, G. Keeping Up with the Emotets: Tracking a Multi-infrastructure Botnet. *Digit. Res. Pract.* **2023**, *4*, 1–29. [CrossRef]
4. Malware Statistics & Trends Report. Available online: https://www.av-test.org/en/statistics/malware/ (accessed on 8 January 2024).
5. Fuji, R.; Usuzaki, S.; Aburada, K.; Yamaba, H.; Katayama, T.; Park, M.; Shiratori, N.; Okazaki, N. Vote-Based Unknown Malware Detection System Using Consortium Blockchain. In Proceedings of the 25-th International Symposium on Artificial Life and Robotics, Beppu, Japan, 22–24 January 2020.
6. Davies, S.R.; Macfarlane, R.; Buchanan, W.J. Majority Voting Ransomware Detection System. *J. Inf. Secur.* **2023**, *14*, 264–293. [CrossRef]
7. Martin, G.; Spencer, D.; Hair, A.; Deepa, K.; Laudanna, S.; Vinod, P.; Visaggio, C.A. Mobile Malware Detection Using Consortium Blockchain. In *Artificial Intelligence for Cybersecurity*; Stamp, M., Visaggio, A.C., Mercaldo, F., Di Troia, F., Eds.; Springer: Cham, Switzerland, 2022; pp. 137–160.
8. Oberheide, J.; Cooke, E.; Jahanian, F. CloudAV: N-Version Antivirus in the Network Cloud. In Proceedings of the 17th Conference on Security Symposium, Berkeley, CA, USA, 28 July–1 August 2008; pp. 91–106.
9. VirusTotal. Available online: https://www.virustotal.com/ (accessed on 8 January 2024).
10. Sathyanarayan, V.S.; Kohli, P.; Bruhadeshwar, B. Signature Generation and Detection of Malware Families. In Proceedings of the Information Security and Privacy 13th Australasian Conference, Wollongong, Australia, 7–9 July 2008; pp. 336–349.
11. Kozachok, A.V. Construction and evaluation of the new heuristic malware detection mechanism based on executable files static analysis. *J. Comput. Hacking Tech.* **2018**, *14*, 225–231. [CrossRef]
12. Nguyen, G.; Nguyen, B.M.; Tran, D.; Hluchy, L. A heuristics approach to mine behavioural data logs in mobile malware detection system. *Data Knowl. Eng.* **2018**, *115*, 129–151. [CrossRef]
13. Jing, C.; Wu, Y.; Cui, C. Ensemble dynamic behavior detection method for adversarial malware. *Future Gener. Comput. Syst.* **2022**, *130*, 193–206. [CrossRef]
14. Liu, S.; Feng, P.; Wang, S.; Sun, K.; Cao, J. Enhancing malware analysis sandboxes with emulated user behavior. *Comput. Secur.* **2022**, *115*, 102613. [CrossRef]
15. Meng, G.; Liu, Y.; Zhang, J.; Pokluda, A.; Boutaba, R. Collaborative Security: A Survey and Taxonomy. *ACM Comput. Surv.* **2015**, *48*, 1–42. [CrossRef]
16. Colajanni, M.; Gozzi, D.; Marchetti, M. Collaborative Architecture for Malware Detection and Analysis. In Proceedings of the IFIP TC 11 23rd International Information Security Conference, Milano, Italy, 7–10 September 2008; pp. 79–93.
17. Marchetti, M.; Messori, M.; Colajanni, M. Peer-to-Peer Architecture for Collaborative Intrusion and Malware Detection on a Large Scale. In Proceedings of the 12th International Conference on Information Security, Pisa, Italy, 7–9 September 2009; pp. 475–490.
18. Bakır, H. VoteDroid: A New Ensemble Voting Classifier for Malware Detection Based on Fine-Tuned Deep Learning Models. *Multimed. Tools Appl.* **2024**, 1–12. [CrossRef]
19. Shahzad, R.; Lavesson, N. Comparative Analysis of Voting Schemes for Ensemble-Based Malware Detection. *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl.* **2013**, *4*, 98–117.
20. Islam, R.; Sayed, M.I.; Saha, S.; Hossain, M.J.; Masud, M.A. Android Malware Classification Using Optimum Feature Selection and Ensemble Machine Learning. *Internet Things-Cyber-Phys. Syst.* **2023**, *3*, 100–111. [CrossRef]
21. Xue, L.; Zhu, T. Hybrid Resampling and Weighted Majority Voting for Multi-Class Anomaly Detection on Imbalanced Malware and Network Traffic Data. *Eng. Appl. Artif. Intell.* **2024**, *128*, 107568. [CrossRef]

22. MalwareBazaar. Available online: https://bazaar.abuse.ch/ (accessed on 8 January 2024).
23. Fung, C.J.; Lam, D.Y.; Boutaba, R. RevMatch: An Efficient and Robust Decision Model for Collaborative Malware Detection. In Proceedings of the Network Operations and Management Symposium, Krakow, Poland, 5–9 May 2014; pp. 1–9.
24. Salem, A.; Banescu, S.; Pretschner, A. Maat: Automatically Analyzing VirusTotal for Accurate Labeling and Effective Malware Detection. *ACM Trans. Priv. Secur.* **2021**, *24*, 1–35. [CrossRef]
25. Zhu, S.; Shi, J.; Yang, L.; Qin, B.; Zhang, Z.; Song, L.; Wang, G. Measuring and Modeling the Label Dynamics of Online Anti-Malware Engines. In Proceedings of the 29th USENIX Conference on Security Symposium, Berkeley, CA, USA, 12–14 August 2020; pp. 2361–2378.
26. Peng, P.; Yang, L.; Song, L.; Wang, G. Opening the Blackbox of VirusTotal: Analyzing Online Phishing Scan Engines. In Proceedings of the Internet Measurement Conference, New York, NY, USA, 21–23 October 2019; pp. 478–485.
27. Cocca, D.; Pirozzi, A.; Visaggio, C. We Cannot Trust in You: A Study about the Dissonance among Anti-Malware Engines. In Proceedings of the 17th International Conference on Availability, Reliability and Security, New York, NY, USA, 23–26 August 2022; pp. 1–13.
28. VirusTotal Stats. Available online: https://www.virustotal.com/gui/stats (accessed on 11 June 2024).