An Experimental Study of Diversity with Off-The-Shelf AntiVirus Engines

Ilir Gashi[1], Vladimir Stankovic[1]
[1]Centre for Software Reliability
City University London
London, UK
{I.Gashi, V.Stankovic}@ city.ac.uk

Corrado Leita[2], Olivier Thonnard[3,4]
[2]Symantec, [3]Royal Military Academy, [4]EURECOM
[2,4]Sophia Antipolis, France, [3]Brussels, Belgium
[2]Corrado_Leita@symantec.com,
[3,4]Olivier.Thonnard@rma.ac.be

Abstract—**Fault tolerance in the form of diverse redundancy is well known to improve the detection rates for both malicious and non-malicious failures. What is of interest to designers of security protection systems are the actual gains in detection rates that they may give. In this paper we provide exploratory analysis of the potential gains in detection capability from using diverse AntiVirus products for the detection of self-propagating malware. The analysis is based on 1599 malware samples collected by the operation of a distributed honeypot deployment over a period of 178 days. We sent these samples to the signature engines of 32 different AntiVirus products taking advantage of the VirusTotal service. The resulting dataset allowed us to perform analysis of the effects of diversity on the detection capability of these components as well as how their detection capability evolves in time.**

Keywords; AntiVirus detection engine analyis, malware detection, cluster analysis

I. INTRODUCTION

All software, including off-the-shelf software, need to be sufficiently reliable and secure in delivering the service that is required of them. There are various ways in which this reliability and security can be achieved in practice, such as the use of various validation and verification techniques in the software construction phases, statistical testing of the final product before delivery, issuance of patches and service releases for the product in operation, as well as the use of software fault/intrusion tolerance techniques. The fault tolerance techniques can range from simple "wrappers" of the software components [1] to the use of diverse software products in a fault-tolerant system [2]. This latter strategy of implementing fault tolerance was historically considered prohibitively expensive, due to the need for development of multiple bespoke software versions. However, the wide proliferation of off-the-shelf software for various applications has made the use of software diversity an affordable option for fault tolerance against either malicious or non-malicious faults.

In the field of intrusion tolerance there have already been examples of implementing intrusion tolerant architectures which employ diverse intrusion detection systems for detecting malicious behaviour [3]. Similarly, a recent publication [4] has also detailed an implementation of an AntiVirus (AV) platform which makes use of diverse AntiVirus products for malware detection. A similar architecture which uses diverse AntiVirus email scanners has been detailed in [5]. Therefore, architectural solutions for employing diverse detection engines (either IDS or AntiVirus products) are already known. Studies which provide an empirical evaluation of the effectiveness of diversity for detection of malware in a rigorous way are, on the other hand, much more scarce.

This paper aims at addressing this research gap by proposing an analysis of the effects of diversity in the most realistic conditions. This is achieved by taking into consideration real-world data generated by a distributed honeypot deployment, SGNET [7], [8]. The advantages of this dataset over those used in the previous work [4] can be summarized as follows:

i. *It is unbiased*. SGNET takes advantage of protocol learning techniques to emulate code injection attacks in a protocol-agnostic way. Differently from other honeypot techniques, SGNET honeypots do not make any assumption on the nature of the exploits to be observed. This maximizes the probability of correctly observing all the threats hitting the honeypot and reduces the bias normally introduced by knowledge-based approaches.

ii. *It emulates realistic conditions*. Whenever a sample is collected by a SGNET honeypot, it is immediately evaluated with the latest version of the AntiVirus signatures. The information at our disposal is thus as near as possible to the security perceived by a real client deployed over the network.

iii. *It provides an evolutionary perspective*. As will be described later, every collected sample is analyzed on a daily basis for at least 30 days with the latest version of the AntiVirus (AV) signatures. This offers perspectives on the window of exposure to the threats, which has been analyzed in this work.

iv. *It is open*. The dataset is freely accessible to any research institution interested in reproducing our results.

The construction of meaningful benchmarks for the evaluation of the detection capability of different AntiVirus products is an open debate in the research community. Modern AntiVirus products consist of a complex architecture of different types of detection components, and achieve better detection capability by combining together the output of these diverse detection techniques. Since some of these detection techniques are also based on analysing the behavioural characteristics of the inspected samples, it is very difficult to set up a benchmark able to fully assess the detection capability of these complex products. Institutions such as the Anti Malware Testing Standards Organization [1] have been specifically generated in an effort to address the issues involved in this task. Moreover, previous research work [9] underlined the challenges in correctly defining the notion of "success" in the detection of a specific malware sample and have raised concerns on the feasibility of generating meaningful benchmarks for this class of intrusion detectors.

On this basis, the evaluation proposed in this paper is defined upon a simplified view of the problem, which we consider sufficient to the accomplishment of our goals. We do the following:

- We take into consideration a single type of component appearing in most AntiVirus products, namely the signature-based detection engine.
- We perform the analysis on unambiguous malicious samples, the samples that are known to be malicious executable files according to the information provided by the SGNET dataset.
- We consider as a successful detection any alarm message provided by the component. We do not try to diagnose the "correctness" of the generated alarm message.

While the resulting measures may not be representative of the detection capability achieved by the real-world operation of the various AV products, they provide an interesting analysis on the detection capability of the respective signature-based sub-components under the above stated conditions.

We take advantage of 1599 malware samples collected by the SGNET honeypot deployment over a period of 178 days in the period February to August 2008. The sample set taken into consideration is thus representative of currently spreading threats. For each malware sample, we study the evolution of the detection capability of the signature-based component of 32 different AntiVirus products and investigate the impact of diversity on such a capability. Through daily analyses of the same sample using the most up-to-date signature database available for each AV product, we have been able to study the evolution of the detection capability over time. We observed that many AntiVirus detection engines give very high detection rates for the analyzed samples, but no sample was correctly identified by all the 32 detection engines. We also found many cases of *regression* in the detection capability of the engines: cases where an engine would go from detecting the malware on a given date to not detecting the same malware at a later date. The results suggest that the use of diverse detection engines developed by different AntiVirus vendors may bring benefits in detection capability.

For the sake of brevity, in the rest of the paper we will use the short-hand notation AV to refer to the signature-based component of an AntiVirus detection engine.

The rest of this paper is organised as follows: section II details the experimental architecture used to collect the data; section III details exploratory empirical analysis of the data collected on the AntiVirus detection behaviour; section IV contains some initial cluster analysis based on the collected dataset; section V reviews two recent implementations that employ diverse AntiVirus engines for detecting malware or scanning potentially malicious emails and section VI contains discussions, conclusions and provisions for further work.

II. Experimental Setup and Architecture

The definition of a representative dataset for the evaluation of the effectiveness of intrusion detection systems is a difficult, if not impossible, task [9]. The malware scenario is evolving continuously, and new threats, or variants of existing threats, are generated on a daily basis.

This work does not aim at being a comprehensive comparison of the detection capability of different products to the variety of Internet threats. We focus on a medium-sized sample set composed of a specific class of threats that are known to be currently active thanks to the observation and collection performed by a distributed honeypot deployment.

The analyzed dataset is composed of 1599 malware samples collected by a real world honeypot deployment, SGNET [7], [8]. SGNET is a distributed honeypot deployment for the observation of server-side code injection attacks. Taking advantage of protocol learning techniques, SGNET is able to fully emulate the attack trace associated to code injection attacks and download malware samples spreading with server-side exploits. By deploying many sensors in different networks of the Internet, SGNET collects in a central repository a snapshot of the aggregated observations of all its sensors. By taking advantage of this data as input to our analysis, we build our analysis upon a limited, but realistic dataset with respect to the modern trends for the specific class of malware (i.e., those associated with code injection attacks).

SGNET information enrichment framework [10] enriches the information collected by the deployment with additional data sources. Two sources are relevant to this work: the behavioural information provided by Anubis[2] [11], [12] and the detection capability information provided by VirusTotal [6].

Every malware sample collected by the deployment is automatically submitted to Anubis to obtain information of its behaviour when executed on a real Windows system. This

[1] www.amtso.org

2 http://anubis.iseclab.org/

information is useful to filter out corrupted samples collected by the deployment, which would not be executable on a real system. Such samples proved to be the cause of ambiguities in the detection capability [9]. It is, however, unclear whether such corrupted samples should or should not be detected: different engines often follow contradicting policies.

The foundations of our analysis are derived from the interaction of SGNET with the VirusTotal service. VirusTotal (VT) is a web service that allows the analysis of a given malware sample by the signature-based engines of different AntiVirus vendors[3]. All the engines are always kept up-to-date with the latest version of the signatures. A submission of a malware sample to VirusTotal at a given point in time thus provides a snapshot on the ability of the different signature-based engines to correctly identify a threat in such samples. It is important to stress that the detection capability evaluation is performed on a subset of the functionalities of the detection solutions provided by the different vendors.

Every time in which a sample is collected by the SGNET deployment it is automatically submitted for analysis to VirusTotal, and the corresponding result is stored within the SGNET dataset. To get information on the evolution of the detection capability of the engines, each sample is resubmitted on a daily basis for a period of at least 30 days. After that, the submission is stopped as soon as the detection results of the last 7 submissions are unchanged. Such a policy allows continuing the submission of the malware, which causes "unstable" detection decisions to be made even 30 days after its collection.

The dataset generated by the SGNET interaction has some important characteristics that need to be taken into account in the following detection capability evaluation.

First, all the malware taken into consideration have been pushed to the victim as a consequence of a successful hijack of its control flow. We can thus safely consider that all the analyzed samples are a result of a malicious and unsolicited activity.

Secondly, all the considered samples are valid Windows Portable Executable files. That is, all these samples run successfully when executed in a Windows operating system.

Thirdly, the definition of difference among two malware samples is based solely on their content. The frequent usage of polymorphic techniques [13] in malware propagation is likely to bias this number. Through polymorphism, a malware modifies its content at every propagation attempt: two instances of the same malware thus appear as different when looking solely at their content. To have an approximate idea of the impact of the phenomenon of the polymorphism on the considered dataset, 1054 of the 1599 samples were injected only once and are thus likely to be instances of polymorphic malware. These 1054 samples could be grouped into 81 different families by looking at their structural characteristics.

Finally, interdependence exists between the submission of a sample to VirusTotal and the observed detection capability. The VirusTotal service actively contributes to the AntiVirus community by sharing with all the vendors all the submitted samples resulting in low detection rate.

3 In our study we evaluated the outputs of 32 AVs.

III. Exploratory Analysis of AV Diversity

We have taken advantage of the previously introduced dataset to perform a detection capability analysis of 32 different AVs when subjected with the 1599 malware samples collected by the SGNET deployment. Exploiting the submission policy implemented in the SGNET dataset, we have considered for each sample the submissions performed on the 30 days succeeding its download. The input to our analysis can thus be considered as a series of triplets, each of which associates a certain malware sample, an AV vendor and the identifier of the submission day with respect to the download date $\{Malware_i, AV_j, Day_k\}$. For each of these triplets we have defined a binary score: 0 in case of successful detection, 1 in case of failure. Table I shows the aggregated counts of the 0s and 1s for the whole period. As previously explained, we have considered as success the generation of an alert regardless of the nature of the alert itself.

TABLE I. THE COUNTS OF SUCCESSFUL DETECTIONS AND FAILURES FOR TRIPLETS $\{MALWARE_I, AV_J, DAY_K\}$

Value	Count
0 – *no failure*	1,093,977
1 – *security / correctness failure*	143,031

For a number of technical reasons in the interaction of the SGNET dataset and VirusTotal a given malware and an AV are not always associated to 30 triplets. In the observation period, some AVs have not been queried on a given day, and some analysis reports have not been stored correctly by VirusTotal.

A. Single AV Product Results

Figure 1 shows the variation of the number of malware that were sent to VirusTotal during the collection period.

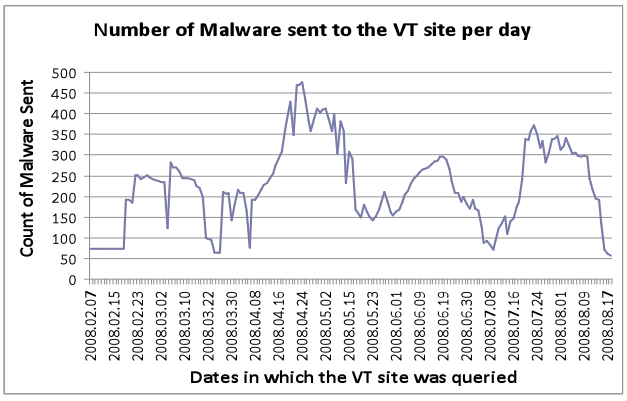

Figure 1. Count of malware sent to the VT site per day

The variation in the number of malware sent can be explained by the relatively small number of malware observed for the initial period, and then by the policy used to send malware (explained in section 2).

Table II lists the top 10 performing AVs ranked by their failure (non-detection) rates[4].

TABLE II. TOP 10 AVS BY THEIR FAILURE (NON-DETECTION) RATES

For all instances of malware		For all distinct malware	
AV Name	***Failure rate***	***AV Name***	***Failure rate***
AV-7	2.7E-04	AV-7	6.3E-04
AV-16	4.5E-04	AV-17	1.3E-03
AV-17	5.9E-04	AV-6	2.5E-03
AV-32	1.0E-03	AV-26	5.0E-03
AV-26	1.5E-03	AV-21	6.3E-03
AV-2	1.5E-03	AV-15	8.8E-03
AV-6	1.8E-03	AV-22	8.8E-03
AV-30	2.5E-03	AV-16	1.0E-02
AV-22	3.2E-03	AV-19	1.0E-02
AV-21	3.2E-03	AV-23	1.0E-02

The two left-most columns list the top 10 AVs and their failure rates for all the instances of the malware sent to them during the collection period: we derive the failure rate by calculating the ratio of the count of failures for all triplets $\{Malware_i, AV_j, Day_k\}$ for a given detection engine AV_j divided by the count of all triplets (failures and successful detections) for the same detection engine AV_j. The right-most two columns show the failure rates per distinct malware: we derive the failure rate by calculating the ratio of the count of distinct malware which were not detected at least once by a given AV_j divided by the count of distinct malware examined by the AV_j (even if the AV_j had failed to detect a given malware only once we would count that as a failure). From the results in Table II we can see that there is substantial variability in the detection capability of the top 10 AVs.

We also found a large number of AVs that somehow fluctuated in their detection capability. In many cases, the AVs regressed in their detection decisions. That is, they detected the malware at first, and then failed to detect the malware at a later date, probably due to some updates in the respective AV's rule definitions. It is interesting to note that a few of the AVs in the Top 10 shown in Table II (five in total from the Top 10 of the right-hand side; and five in total from the left-hand side (with three in both lists)) are among the AVs that regressed. Table III shows the top 10 worst performing AVs in terms of the number of these regression failures. The second column shows the count of distinct malware which they detected at first, but failed to detect in a subsequent submission to the VT site during our collection period. The third column shows the count of the triplets $\{Malware_i, AV_j, Day_k\}$ for which a given AV_j regressed.

Results presented in Table III underline an interesting behavior: we have observed patterns characterizing several AVs on several malware where the AVs switch from *detecting* to *not detecting* a given malware multiple times in our collection period. This would indicate that the updated rule set of a given AV breaks its detection capability making the AV fail to detect a malware that was previously detected correctly. Such a phenomenon can be due to various reasons. For instance, the vendor might have deleted the corresponding detection signature as a consequence of the identification of false positives associated to it.

TABLE III. THE WORST 10 AVS BY THE COUNT OF DISTINCT MALWARE, AND THE COUNT OF $\{MALWARE_i, AV_J, DAY_K\}$ TRIPLETS ON WHICH THEY REGRESS

AV Name	Number of Malware the AV regressed on	Number of instances the AV regressed on
AV-20	586	1691
AV-8	374	538
AV-3	71	78
AV-11	58	59
AV-1	37	235
AV-32	36	38
AV-2	15	15
AV-16	13	13
AV-9	8	8
AV-14	4	4

Figure 2 illustrates an example of a repeated regression of two AVs (one is commercial, the other is open-source) on a single malware.

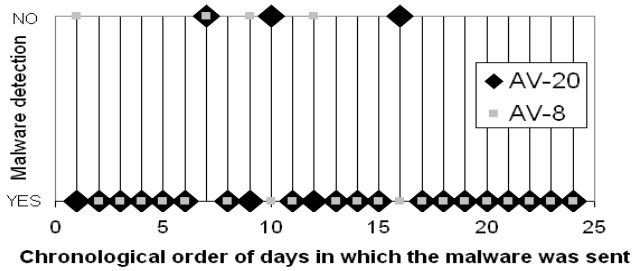

Figure 2. The detection capability of two AVs on one malware during the collection period

We can see that even though the two AVs detect the malware on most days, they do not detect it every day. We can also observe that both of the AVs fail to detect the same malware on the same day *only* once, on the 7th day: in other days when one AV fails the other does not. This would indicate gains from employing diverse AV engines.

Figure 3 shows the 1599 malware observed in our collections period, and counts of how many AVs failed to detect a given malware at least once.

We can see that the most "difficult" malware was not detected by 21 AVs in our study. There were no malware which caused all the different AVs to fail and only one malware which was detected by all the AVs.

[4] The AV names have been anonymised to prevent concerns deriving from the comparison of commercial products.

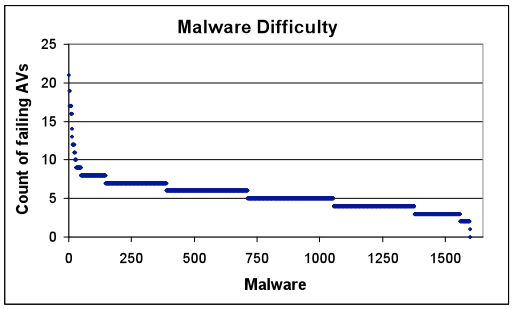

Figure 3. Malware difficulty

Figure 4 shows a contour plot of the data collected in our study. The x-axis lists all 32 AVs ordered by their lowest to highest failure rates (left to right). The y-axis lists all 1599 malware ordered by their detection difficulty – the number of AVs that have failed to detect it. The malware are sorted from the easiest to the most difficult one, on average for all AVs (bottom to top). Each cell in the contour plot shows a failure rate of a given AV on a given malware throughout the collection period. The failure rate values are represented with the specific colouring schema (see the color bar on the right hand side of the figure). The values are in the range 0 (represented with a shade of gray) to 1 (represented in white color). The contour plot, however, includes also the values of -1, which are drawn in black color and represent the cases where "no result" exists, i.e., the cases where the malware was not sent to a given AV at all during the collection period. A failure rate value of, for example, 0.2 in a given cell is represented with a shade of gray color and can be interpreted as follows: "AV *i* failed to detect the malware *j* on 20% of the days during our collection period on which the malware was submitted to the AV".

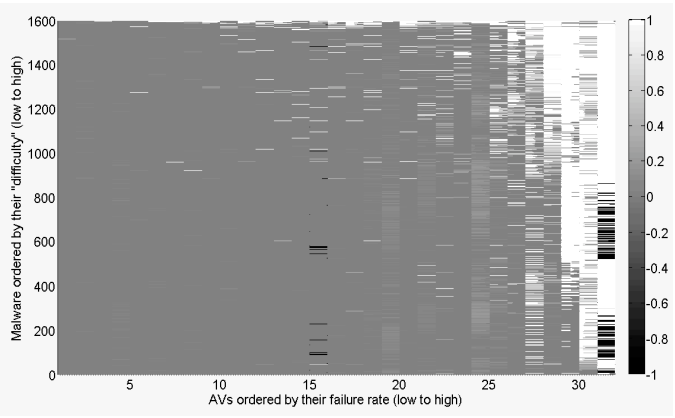

Figure 4. A contour plot of the AV (x-axis) failure rates (z-axis, represented by the intensity of the colour in the plot) over the malware (y-axis)

The bottom left side of the plot shows the best performing AVs on the "easy" malware (on average) whereas the top right corner shows the worst performing AVs on the "difficult" malware (on average). We can also see many examples of "white" horizontal lines (difficult demands) for one AV, which are "shaded gray" lines for many of the other AVs. This would indicate potential benefits of employing diverse AVs.

B. Results with 1oo2 Pairs

We saw in the previous sub-section the potential benefits in malware detection that may be obtained from employing diversity. In this sub-section we will present in more detail results that give initial quantifications of these gains. We will base our analysis on pairs of AVs constructed from the 32 AVs used in our study.

Table IV shows the count of 1oo2 ("1-out-of-2") pairs[5] and single versions within a given band of failure rate. We can see that over 18% of the 1oo2 pairs achieve perfect detection. A total of 163 1oo2 pairs out of 496 (32.9%) have a better detection rate than the best performing single AV (i.e., 163 pairs have a failure rate lower than 2.7E-04).

TABLE IV. COUNTS OF SINGLE AVS AND 1OO2 PAIRS PER FAILURE RATE BAND

Failure rate (f.r.)	Count (and % of the total) of 1oo2 pairs	Count (and % of the total) of single AVs
failure rate = 0	91 (18.35%)	0 (0%)
1.0E-05 ≤ f. r. <1.0E-04	43 (8.67%)	0 (0%)
1.0E-04 ≤ f. r. <1.0E-03	111 (22.38%)	3 (9.37%)
1.0E-03 ≤ f. r. <1.0E-02	200 (40.32%)	13 (40.63%)
1.0E-02 ≤ f. r. <1.0E-01	37 (7.46%)	10 (31.25%)
1.0E-01 ≤ f. r. <1.0	14 (2.82 %)	6 (18.75%)
Total pairs	**496**	**32**

A user of an AV may also be interested in the gains that a specific 1oo2 configuration of AV engines will have over the individual AVs. The user may be running an AV engine, say AV_i, in their system, and be interested in the potential gains in detection capability (in terms of lower failure / non-detection rate) from running alongside it another AntiVirus engine, say AV_j.

Figure 5 presents this evidence for the 496 pairs ($^{32}C_2$ combinations) in our study. The x-axis lists the 496 1oo2 pairs. The y-axis lists the potential gains in detection capability (i.e., lower failure rate) from switching from a single AV to a 1oo2 pair (given in a logarithmic scale 0 to 1). The dots show the gains of switching from the AV with the worse failure rate to the corresponding 1oo2 pair: this value was calculated by subtracting the failure rate value of the 1oo2 pair from the AV with worse (i.e., higher) failure rate; whereas the solid line represents the gains of switching from the AV with the better failure rate to the 1oo2 pair (hence the gains in this latter case are always lower). We can see that for a lot of the pairs the maximum gains can be quite high (failure rates of the 1oo2 pairs decrease almost 100%). The solid line values fall to zero in the bottom right-hand side of the graph. This indicates that the pair is made up of an AV, say AV1, which only fails for a subset of the demands (i.e., triplets {$Malware_i$, AV_j, Day_k}) that the other AV, say AV2, fails on, and no other. Hence AV1 will see no improvement when paired with AV2.

[5] A configuration where the system will detect the malware as long as one out of the two AVs in the diverse setup detects the malware.

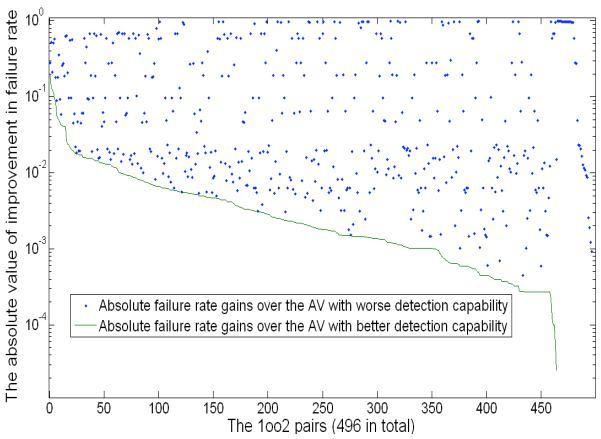

Figure 5. Gains in detection capability (lower failure rates) of a pair of AVs over the individual AVs.

IV. Cluster Analysis of AV Detection Capability

As the results in the previous section suggest, certain combinations of AVs provide better detection capabilities, and hence improved security. So this section presents a cluster analysis performed on the failure rates of all AVs and AV pairs, as calculated from our malware dataset. The objective is to visualize how similarly some AVs, as well as AV pairs, performed on this dataset. More importantly, we want to discover the presence (or absence) of relationships between AV clusters and the detection rate resulting from their combination.

For each AV, we have calculated the daily normalized failure rate by computing the ratio of the number of non-detections (failures) against the total number of samples submitted on each day. This daily failure rate is represented by a vector of 178 elements, i.e., one element per day, which can then be used as *feature vector* by a clustering algorithm. When dealing with such multi-dimensional data, it is often useful to see them charted on a two-dimensional map in order to understand the relationships between those data items. Multidimensional scaling (MDS) is a set of methods that address this type of problem. MDS is based on dimensionality reduction techniques, which aim at converting a multi-dimensional dataset into a two or three-dimensional representation that can be displayed e.g., in a scatter plot. As a result, MDS allow an analyst to visualize how near observations are to each other for many kinds of distance or dissimilarity measures, which in turn can deliver insights into the underlying structure of the high-dimensional dataset.

Because of the non-linear characteristic of failure rate vectors, we applied a recent MDS technique called *t-SNE* to visualize our dataset. t-SNE [14] is a variation of *Stochastic Neighbour Embedding (SNE)* and it produces significantly better visualizations than other MDS techniques by reducing the tendency to crowd points together in the centre of the map. Moreover, this technique has proven to perform better in retaining both the local and the global structure of real, high-dimensional data in a single map, in comparison to other non linear dimensionality reduction techniques such as Sammon mapping, Isomaps or Laplacian Eigenmaps.

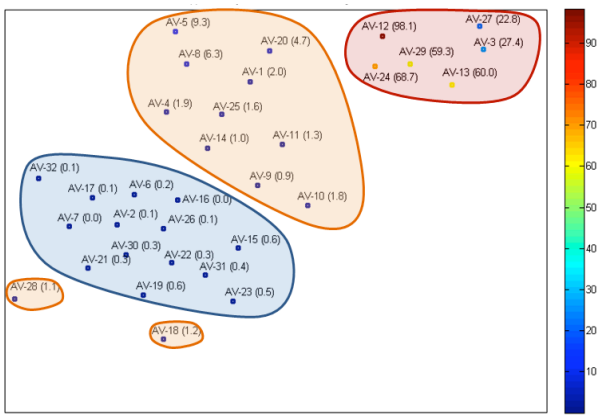

Figure 6. Visualizing failure rates (by date) for single AVs using Multidimensional Scaling

Figure 6 shows the resulting two-dimensional plot obtained by mapping the failure rates of all 32 AVs over the whole collection period. We can identify three clusters of AVs that exhibit similar detection capabilities:

- a group of 6 AVs with low detection capabilities (in the upper right corner) whose failure rate lies (on average) between 22.8% and 98.1%;
- a group of 14 AVs with high detection capabilities (in the lower left part) with a failure rate under 0.6%; and
- a group of 10 AVs (+ 2 outliers) which had an intermediate detection capability (a failure rate between 1.0% and 9.3%). The colouring of each data point indicates the corresponding mean failure rate (over the whole period).

In section III we saw the potential benefits in malware detection that may be obtained from employing diversity, based on the analysis of AV pairs deployed in a 1oo2 configuration. We can again apply t-SNE to create a low-dimensional representation of the failure rate vectors corresponding to all 496 possible AV pairs, which leads to a global visualization of the results given previously in the second column of Table IV. Each data point on the plot has been mapped from the normalized failure rate vector (178 elements, i.e., one per day) of a given AV pair, and the colouring refers to the 5 failure rate intervals as defined in Table IV (with 0 corresponding to a zero failure rate and 5 corresponding to failure rate band 1.0E-01 to 1.0).

In Figure 7, in the left part of the map (blue circle), we can clearly see one large cluster of 91 AV pairs whose failure rate was equal to zero (i.e., perfect detection). In the top right corner (red circle), we see a smaller cluster of 14 AV pairs whose average failure rate lies between 11% and 68% (i.e., low performing AV pairs).

A further exploration of this two-dimensional representation of AV pairs delivers some interesting results. Let us consider the following AV clusters as identified previously from Figure 6:

- The cluster of 6 AVs with low detection capabilities C1 = {AV-12, AV-24, AV-13, AV-29, AV-3, AV-27}

- The cluster of 14 AVs with high detection capabilities C2 = {AV-7, AV-16, AV-26, AV-17, AV-32, AV-2, AV-6, AV-21, AV-22, AV-30, AV-31, AV-23, AV-15, AV-19};

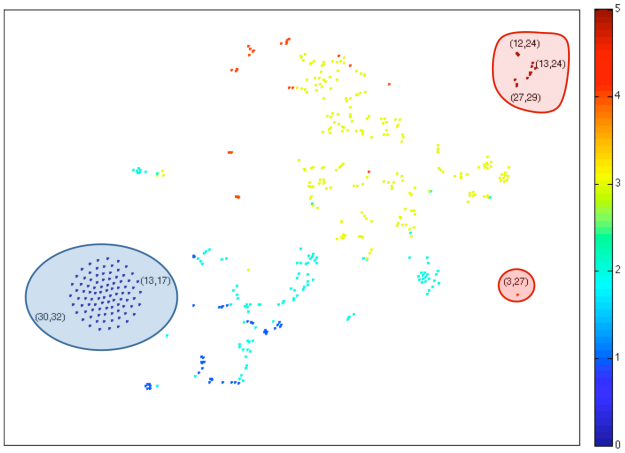

Figure 7. Visualizing failure rates of AV pairs using Multidimensional Scaling

Taking advantage of this clustering technique we can infer interesting facts on the relationship between the AV clusters introduced in Figure 6 and the detection rate deriving from their combination:

- *Combination of [Low+High] performing AVs*: Out of 84 possible pairs, there are 15 cases (18%) that have a significant improvement in detection rate, of which 11 cases show even a perfect detection rate (for example: the pair [AV-13, AV-17]). None of those 84 combinations were found in the two highest failure rate bands (i.e., intervals nr 4 and 5), which means that combining those low and high performing AVs has always delivered very high gains in detection capability during this experiment.
- *Combination of [Low+Low] performing AVs*: There are 15 possible combinations, among which 14 are situated in the worst detection capability group (i.e., the band with the highest failure rate), which means we obtain a marginal gain combining the products from the cluster of "low-performing" AVs. Only one AV pair shows some minor improvement ([AV-3, AV-27]) with a failure rate of 9%, but it is still performing quite poorly.
- *Combination of [High+High] performing AVs*: Among the 91 possible combinations, 63 pairs (69%) have a significant improvement of the failure rate with respect to the single version, of which 46 pairs belong to the cluster of AVs with perfect detection rate (FR=0). Only 10 AV pairs of this type have somewhat a marginal performance gain, hence situated a bit more in the middle of the map (in the failure rate band nr 3).

Obviously, the diversity obtained by combining two high performing AVs delivers almost always a higher gain in detection capability, but this can also have a higher impact on the total system cost, since a majority of the AVs with high detection rates are commercial ones (only 2 out of 14 AVs belonging to cluster C2 are free AVs). From the results given above, it seems that the combination of a pair of AVs, which individually have contrasting detection capabilities, is a better trade-off between an improved detection capability of the system and its total cost of ownership.

Interestingly, one pair of free AVs was even found among the group of AV pairs with perfect detection rate, and four other pairs of free AVs were found in the first failure rate band (i.e., between 1.0E-5 and 1.0E-4), which indicates that high detection rates could be achieved even with pairs of free AV products.

V. Related Work on Architectures that Utilise Diverse AntiVirus Products

We have so far presented the potential gains in detection capability that can be achieved by using diverse AVs. We have addressed a complex problem, that of generating a meaningful evaluation of AV detection techniques, by self-imposing a set of limitations in the notion of detection capability of the AVs. Within these bounds, we have attempted to maximize the realism of the experimentation taking advantage of the SGNET deployment.

While these efforts are, to the best of our knowledge, a novel contribution introduced by this work, the concept of combining multiple detectors is not new and was proposed in recent publications. Oberheide et al. in [4] have proposed an architecture called Cloud-AV, which utilises multiple diverse AV products to improve the detection performance. The Cloud-AV architecture is based on a client-server paradigm, in which each client submits suspicious files to a central network service in charge of combining the results obtained by the operation of multiple AntiVirus products. Consistently with this work, the authors show in [4] how the employment of diversity can lead to considerable advantages in performance. The experimentation carried on in [4] is solely instrumental to the validation of the idea, and can be considered complimentary to the more extensive one carried out in this work.

Finally, another implementation [5] is a commercial solution for e-mail scanning which utilises diverse AntiVirus engines.

VI. Discussion and Conclusions

In this paper we presented analysis of the potential gains in reliability (detection rates) that can be obtained from using more than one diverse AntiVirus signature-based detection engine. We tested 32 engines hosted by the VirusTotal site [6] with 1599 malware samples collected from a distributed honeypot platform. The malware were observed in the honeypots in a six month period between February and August 2008.

The analysis proposed in this work is an assessment of the practical impacts of the application of diversity in a real world scenario based on realistic data generated by a distributed honeypot deployment. As shown in [9], the comprehensive

evaluation of detection capability of AntiVirus engines is an extremely challenging, if not impossible, problem. This work does not aim at providing a solution to this challenge, but builds upon it to clearly define the limits of validity of its measurements.

The detection capability analysis of the signature-based components showed a considerable variability in detection of the malware samples considered in the dataset. Also, despite the generally high detection rate of the detection engines, none of them achieved 100% detection rate. The detection failures were both due to the lack of knowledge of a given malware at the time in which the samples were first detected, but also due to regressions in the ability to detect previously known samples as a consequence, possibly, of the deletion of some signatures.

The differences in performance of the AVs justified the use of diversity for improving the detection capability. From our experiments with 1-out-of-2 (1oo2) pairs of engines, the improvements in detection capability resulting from the usage of diversity are significant. Almost a third of the resulting pairs achieved a better detection rate compared with the best individual engine. Interestingly, we saw how one pair constructed from free detection engines had a perfect detection rate for the malware collected in our study and four other pairs of free detection engines had higher detection capability than the best individual engine.

There are several provisions for further work:

i) As we stated in the introduction, there are many difficulties with constructing meaningful benchmarks for the evaluation of the detection capability of different AntiVirus products (see [9] for a more elaborate discussion). Modern AntiVirus products comprise a complex architecture of different types of detection components, and achieve higher detection capability by combining together the output of these diverse detection techniques. Since some of these detection techniques are also based on analysing the behavioural characteristics of the inspected samples, it is very difficult to setup a benchmark able to fully assess the detection capability of these complex components. In our study we have concentrated on one specific part of these products, namely their signature-based detection engine. Further studies are needed to test the detection capabilities of these products in full.

ii) Studying the detection capability with different categories of malicious files. In our study we have concentrated on malicious executable files only. Further studies are needed to check the detection capability for other types of files e.g., document files, media files etc.

iii) Analysis of the benefits of diversity when more than two AV products are used, such as 2oo3 ("two-out-of-three") diverse configuration, or configurations with a higher number of diverse AV products.

ACKNOWLEDGMENT

This work has been supported by the European Union Framework Programme 6 via the "Resilience for Survivability in Information Society Technologies" (ReSIST) Network of Excellence (contract IST-4-026764-NOE). This work has also been partially supported by the European Commission through project FP7-ICT-216026-WOMBAT funded by the 7th framework program. The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the European Commission.

We would like to thank Bev Littlewood for reviewing an earlier draft of this paper.

REFERENCES

[1] van der Meulen, M.J.P., et al. *Protective Wrapping of Off-the-Shelf Components*. in *the 4th International Conference on COTS-Based Software Systems (ICCBSS '05)*. 2005. Bilbao, Spain: Springer.

[2] Strigini, L., *Fault Tolerance Against Design Faults*, in *Dependable Computing Systems: Paradigms, Performance Issues, and Applications*, H. Diab and A. Zomaya, Editors. 2005, J. Wiley & Sons. p. 213-241.

[3] Reynolds, J., et al. *The Design and Implementation of an Intrusion Tolerant System*. in *Dependable Systems and Networks (DSN-02)*. 2002. Washington, D.C., USA: IEEE Computer Society Press.

[4] Oberheide, J., E. Cooke, and F. Jahanian. *CloudAV: N-Version Antivirus in the Network Cloud*. in *Proceedings of the 17th USENIX Security Symposium*. 2008.

[5] GFi. *GFiMailDefence Suite*. 2008 last accessed 2008; Available from: http://www.gfi.com/maildefense/.

[6] [6] VirusTotal. *VirusTotal - A Service for Analysing Suspicious Files*. 2008 last accessed 2008; Available from: http://www.virustotal.com/sobre.html.

[7] Leita, C., *SGNET: Automated Protocol Learning for the Observation of Malicious Threats*, in *Institute EURECOM*. 2008, University of Nice: Nice, France.

[8] Leita, C. and M. Dacier. *SGNET: A Worldwide Deployable Framework to Support the Analysis of Malware Threat Models*. in *Seventh European Dependable Computing Conference (EDCC 2008)*. 2008. Kaunas, Lithuania.

[9] Leita, C., et al. *Large Scale Malware Collection: Lessons Learned*. in *Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems, 27th International Symposium on Reliable Distributed Systems (SRDS 2008)*. 2008. Napoli, Italy.

[10] Leita, C. and M. Dacier. *SGNET: Implementation Insights*. in *IEEE/IFIP Network Operations and Management Symposium (NOMS 2008)*. 2008. Salvador da Bahia, Brazil.

[11] Bayer, U., C. Kruegel, and E. Kirda, *TTAnalyze: A Tool for Analyzing Malware*, in *Information Systems Institute*. 2005, Technical University of Vienna: Vienna, Austria. p. 86.

[12] Bayer, U., et al., *Dynamic Analysis of Malicious Code.* Journal in Computer Virology, 2006. **2**(1): p. 67–77.

[13] F-Secure. *Malware Information Pages: Allaple.A*. 2007 last accessed 04 May 2007; Available from: www.f-secure.com/v-descs/allaplea.shtml.

[14] van der Maaten, L.J.P. and G.E. Hinton, *Visualizing Data Using t-SNE.* Journal of Machine Learning Research, 2008. **9**: p. 2579-2605.