Diverse Protection Systems for Improving Security: a Study with AntiVirus Engines

Peter Bishop, Robin Bloomfield, Ilir Gashi, and Vladimir Stankovic

Abstract—Diverse "barriers" or "protection systems" are very common in many industries, especially in safety-critical ones where the designers must use "defense in depth" techniques to prevent safety failures. Similar techniques are also commonly prescribed for security systems: using multiple, diverse detection systems to prevent security breaches. However empirical evidence of the effectiveness of diversity is rare. We present results of an empirical study which uses a large-scale dataset to assess the benefits of diversity with an important category of security systems: AntiVirus products. The analysis was based on 1599 malware samples collected from a distributed honeypot deployment over a period of 178 days. The malware samples were sent to the signature engines of 32 different AntiVirus products hosted by the VirusTotal service. We also present an exploratory model which shows that the number of diverse protection layers that are needed to achieve "perfect" detection with our dataset follows an exponential power-law distribution. If this distribution is shown to be generic with other datasets, it would be a cost-effective means for predicting the probability of perfect detection for systems that use a large number of barriers based on measurements made with systems that are composed of fewer (say 2, 3) barriers.

Index Terms— Fault-tolerance, Security and Privacy Protection, Security assessment, Anti-virus engines, Empirical assessment.

1 Introduction

ALL systems, including those built from off-the-shelf components, 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: use of various validation and verification techniques in the software construction phases, issuance of patches and service releases for the product in operation, as well as the use of software fault/intrusion tolerance techniques. 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 developing multiple bespoke software versions. However, the wide proliferation of off-the-shelf software for various application domains has made software diversity an affordable option for fault tolerance against either malicious or non-malicious faults.

Intrusion-tolerant architectures that employ diverse intrusion detection systems for detecting malicious behavior have been proposed in the past [3]. A more recent publication [4] has also detailed an implementation of an AntiVirus platform that makes use of diverse AntiVirus products for malware detection. Similar architectures that use diverse AntiVirus engines for file and email scanning have been commercially available for several years [5-7]. Therefore, architectural solutions for employing diverse detection engines (either IDS or AntiVirus products) are already known and in some cases commercially deployed. Studies that provide empirical evaluation of the effectiveness of diversity for detection of malware and intrusions are, on the other hand, much more scarce.

The following claim is made on the VirusTotal site [8], [9]: "Currently there is not any solution which provides 100% detection rate for detecting viruses and malware". Given these limitations of individual AntiVirus engines, designers of security protection systems are interested in at least getting estimates of what the possible gains are in terms of added security that the use of diversity (e.g. diverse AntiVirus products) may bring for their systems.

In this paper we aim to address this research gap. We performed an analysis of the effects of diversity taking advantage of real-world data, namely the information provided by a distributed honeypot deployment, SGNET [10], [11]. We analyzed 1599 malware samples collected by the SGNET distributed honeypot deployment over a period of 178 days between February and August 2008. Using these malware samples, we studied the evolution of the detection capability of the signature-based component of 32 different AntiVirus products and investigated the impact of diversity on such a capability. Through daily analyses of the same sample by the most up-to-date signature database available for each AntiVirus product, we were able to study the evolution of the detection capability over time.

Utilizing this dataset, we analyzed the detection capabilities of different AntiVirus detection engines and potential improvements in detection that can be observed from using diverse AntiVirus detection engines. We observed that some AntiVirus products achieved high detection rates, but none detected all the malware samples in our study.

We have quantified the possible gains in malware detec-

• P. Bishop and R. Bloomfield are with the Centre for Software Reliability, City University London, EC1V 0HB, UK, and the Adelard LLP, London EC1R 0JH, UK. E-mail: {pgb, reb}@adelard.com.
• I. Gashi and V. Stankovic are with the Centre for Software Reliability, City University London, EC1V 0HB, UK. E-mail:{i.gashi, v.stankovic}@csr.city.ac.uk.

Manuscript received (insert date of submission if desired). Please note that all acknowledgments should be placed at the end of the paper, before the bibliography.

tion from using more than one diverse engine. We have done this empirical analysis for several types of diverse setups ranging from *simple detection setups* (where a malware is deemed to have been detected as soon as one of the AntiVirus products raises an alarm for it) to *majority voting setups* (where a malware is deemed to have been detected only if a majority of AntiVirus products in a given diverse configuration raise an alarm for that malware).

We also analyzed the dataset in the time dimension to quantify the extent to which the use of diverse AntiVirus engines reduces the "at risk time" of a system.

Finally we observe that an exponential power law model seems a good fit for the probability of observing non-perfect detection systems (those with a failure rate greater than zero) as we add more diverse AntiVirus products.

We have reported on some of these results in the past [12], [13], but in this paper we present a consolidated and comprehensive analysis of the effects of various diverse AntiVirus configurations, incorporating new empirical analysis for diverse setups we have not explored in the previous work (such as majority voting setups) and further insights on the modeling aspects of the benefits of diversity.

The analysis in this paper is based upon a simplified view, 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, i.e. 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 findings may not be representative of the full detection capability achieved by the real-world operation of the various AntiVirus products, they provide an interesting analysis of the detection capability of the respective signature-based components under the stated conditions. Also, the purpose of our study is not to rank the individual AntiVirus engines, but to analyze the benefits of diversity from improved detection rates of using more than one AntiVirus product.

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 organized as follows: section 2 details the experimental architecture used to collect the data; section 3 details empirical analysis of the results obtained for the single AVs; Section 4 presents the analysis of benefits of diversity with AntiVirus products. Section 5 presents an analysis of the detection capability of the AVs along the time dimension. Section 6 provides details of an exponential power law model, which appears to be a good fit to the probability of having a system with an observed zero failure rate when we increase the number of AVs in a diverse system; section 7 presents a discussion of the results and limitations on the claims we can make about the benefits of diversity and the limitations of the dataset we have used; section 8 reviews two recent implementations that employ diverse AntiVirus engines for detecting malware or scanning malicious emails and also discusses other related work; section 9 contains conclusions and finally section 10 contains provisions for further work.

2 Experimental Setup and Architecture

The construction of meaningful benchmarks for the evaluation of the detection capability of different AntiVirus products is an open debate in the research community. Previous work [14] underlined the challenges in correctly defining the notion of "success" in the detection of a specific malware sample. Also, modern AntiVirus products consist of a complex architecture of different types of detection components, and achieve higher performance by combining together the output of these diverse detection techniques. Since some of these detection techniques are also based on analyzing 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.

This work does not aim at being a comprehensive benchmark of the detection capability of different products to the variety of Internet threats. Instead, we focus on a medium-sized sample set composed of a specific class of threats.

The analyzed dataset is composed of 1599 malware samples collected by a real world honeypot deployment, SGNET [10, 11]. 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 with code injection attacks and download malware samples that spread using server-side exploits. By deploying many sensors in different networks of the Internet, SGNET collects in a central dataset a snapshot of the aggregated observations of all its sensors. We use this data as input to our analysis and we build our analysis upon a limited, but realistic dataset with respect to the modern trends for a specific class of malware (i.e. malware associated with code injection attacks).

The SGNET information enrichment framework [14] enriches the information collected by the deployment with additional data sources. Two sources are relevant to this work: the behavioural information provided by Anubis [15, 16] and the detection capability information provided by VirusTotal [8].

Every malware sample collected by the deployment is automatically submitted to Anubis to obtain information of its behavior once executed on a real Windows system. This 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 [14]: it is unclear whether such corrupted samples should or should not be detected since different engines often follow conflicting policies.

The foundations of our analysis are derived from the interaction of SGNET with the VirusTotal service. VirusTotal is a web service that allows the analysis of a given malware sample by the signature-based engines of different AntiVirus vendors. All the engines are kept up-to-date with the latest version of the signatures. Thus, a submission of a malware sample to VirusTotal at a given point in time 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 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, for this dataset, is resubmitted on a daily basis for a period of 30 days.

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

Firstly, 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. All these samples run successfully when executed against a Windows operating system.

Thirdly, the malware samples are differentiated based solely on their content. Thus, the frequent usage of polymorphic techniques (an example of which is given in [17]) in malware propagation is likely to bias the number of malware samples collected. 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.

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 improved detection rates across different AV engines.

3 Exploratory Analysis of Single AV Detection Performance

We use the dataset introduced in the previous section 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 associating together a certain malware sample, an AV product and the identifier of the submission day with respect to the download date {AV_i, $Malware_j$, 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 1 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.

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 we have some missing data since some AVs have not been queried on a given day.

Table 1. The Counts Of Successful Detections And Failures For Triplets { AV_I, $Malware_J$, Day_K}

Value	Count
0 – no failure / detection	1,093,977
1 – failure / no detection	143,031

3.1 Summary of Single AV Results

Table 2 lists the top 10 performing AVs[1] ranked by their observed failure (non-detection) rates[2]. The difference between the failure rate values in the second column and in the fourth column is dependent on the definition of a unique "demand". For the failure rates calculated in the second column, a unique demand is a {$Malware_i$, Day_k} pair, i.e. each malware sent on a different day is treated as unique. Hence the maximum number of demands is the product of the number of distinct malware (1599) and the number of days that an AV product is sent this malware (maximum of 30 days), i.e. 1599 * 30 = 47,970.

For the failure rates calculated in the fourth column, unique demands are the 1599 malware samples. In this case, a successful detection by an AV product AV_i of a given malware sample $Malware_i$ happens when AV_i successfully detects $Malware_i$ on all the dates in which it has inspected this malware.

From the results in Table 2 we can see that there is substantial variability in the detection capability of the top 10 AVs.

Figure 1 shows a 3-dimensional 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 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 coloring 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 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 fol-

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

[2] Any mention of "failure rates" in this section refers to the observed (empirical) failure rates calculated from our dataset.

TABLE 2. 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

lows: "*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*".

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.

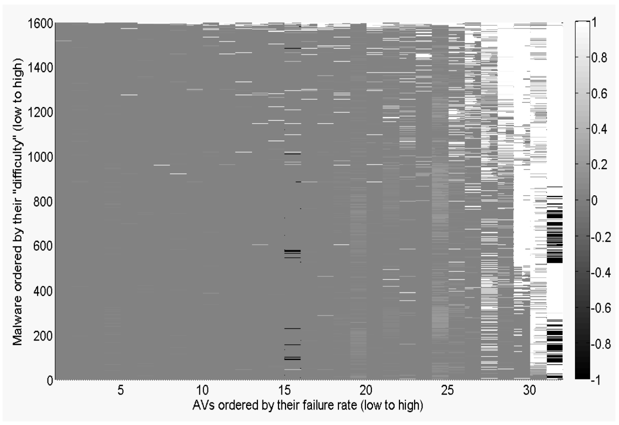
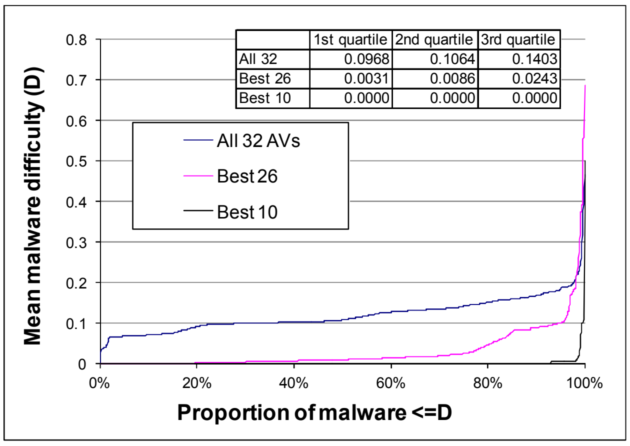


Fig. 1. A plot of the AV (x-axis) failure rates (z-axis, represented by the intensity of the color in the plot) over the malware (y-axis).

The most "difficult" malware, i.e. the ones that fail to be detected by most AVs, appear in the top part of Figure 1. Figure 2 illustrates better this malware difficulty. It shows the mean malware difficulty for all the 1,599 malware samples in our study. The mean malware difficulty is calculated by averaging over all the dates and AVs on which a given malware has been inspected (i.e. a ratio of all the "1s" over the sum of "0s" and "1s" (cf. Table 1), for any given malware). So, it is the marginal distribution when summing over the malware. Figure 3 shows how this difficulty changes when we calculate the averages for all 32AVs, the best 26AVs (removing the six worst AVs which from Figure 2 appear to have very low detection rates) or the best 10 AVs (ranked over their detection rate). The x-axis shows the proportion of malware less than or equal to a given difficulty.

The difficulty distribution gets more extreme as we use



	1st quartile	2nd quartile	3rd quartile
All 32	0.0968	0.1064	0.1403
Best 26	0.0031	0.0086	0.0243
Best 10	0.0000	0.0000	0.0000

Fig. 2. Malware difficulty for different sets of AVs, and the respective quartiles.

better (on average) subsets of AVs. The detection performance is better for most malware when we remove the poorly performing AVs from the calculations of average difficulty, but there are some malware that are hard to detect for all AV sets, so the maximum difficulty remains much the same (does not fall below 0.5).

While better subsets of AVs have higher detection rates on average for all malware, a better subset of AVs can have a worse detection rate on average for a specific malware. This is illustrated in Figure 3 where we show the mean difficulty of the malware for different AV subsets ordered by the mean difficulty for the complete set of AVs (all 32). While difficulty ordering is not preserved for different AV sets, there is a generally increasing trend. In particular there are some malware at the right hand edge that seem to be equally difficult for the best 10 AVs as they are for all of them.

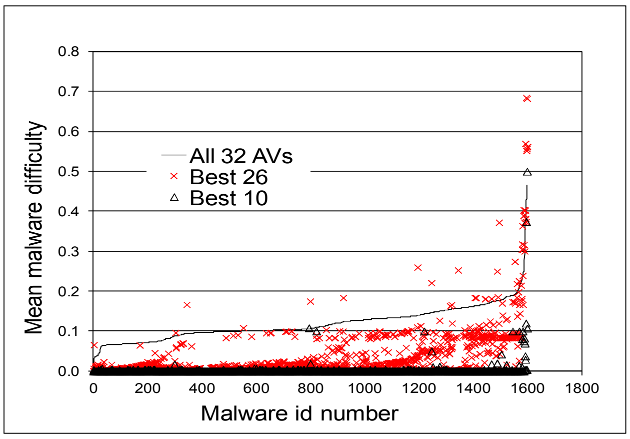


Fig. 3. Malware difficulty for different sets of AVs.

Table 3 shows the count of single versions within a given band of failure rate when we calculate their average failure rate for all instances of malware in our dataset (47,970 demands).

From both Table 3 and Figure 1, we see that 6 AV products are performing very badly for this dataset: they fail for more than 10% of all the demands sent to them. It is inconceivable that any system administrator would choose AVs that have such a bad detection rate to be used in their system. Hence we decided to remove the 6 worst performing AVs from further analysis. The decision was also influenced by our goal to make any results of the benefits of diversity appear fairer. A criticism that can be made if these 6 worst performing AVs are included in the analysis is that improvements in detection capability through the use of diversity will of course be higher if you have such poorly performing individual AVs in the mix. By removing them we make our estimates of the benefits of diversity more conservative.

TABLE 3. COUNTS OF SINGLE AVS PER FAILURE RATE BAND

Failure rate (f.r.)	**Count (and % of the total) of single AVs**
failure rate = 0	0 (0%)
1.0E-05 ≤ f. r. <1.0E-04	0 (0%)
1.0E-04 ≤ f. r. <1.0E-03	3 (9.37%)
1.0E-03 ≤ f. r. <1.0E-02	13 (40.63%)
1.0E-02 ≤ f. r. <1.0E-01	10 (31.25%)
1.0E-01 ≤ f. r. <1.0	6 (18.75%)
Total single AVs	**32**

4. DIVERSE AV DETECTION PERFORMANCE

We now look at the benefits that using more than one AV may bring in terms of detection behavior. The analysis is based on the 26 top-performing individual AV products. The observed failure rates are calculated from a maximum of 47,970 demands, as explained in the previous section.

We have looked at two different types of diversity configurations:

- ***1-out-of-N*** (abbreviated **1ooN** in the rest of the paper), where N is the total number of AV products in a given configuration – in this configuration a malware is deemed to have been detected on a given date as long as *at least one* of the AV products out of *N* in a given configuration detected that malware on that date. These configurations might be especially useful for systems where the administrators wish to be extra cautious and raise an alarm about a malware as soon as one of the AVs in a given configuration has done so.
- ***r-out-of-N*** (abbreviated **rooN** in the rest of the paper) where *N* is the total number of AV products in a given configuration and *r* is the minimum number of AVs that should detect a malware on a given date for this malware to be deemed as detected. These configurations might be especially useful for systems where the administrators are worried about a high rate of false alarms, and wish to only raise an alarm if *r out of* N AV products in a given configuration do so. In our experiments we looked at two particular voting options:
  - o *majority voting:* where *N* is an *odd number* and *r* is *(N+1)/2* - this allows us to have a tie-breaker via *majority voting;*
  - o *trigger level:* where a fixed number *r* of AVs have to detect a malware (i.e. "be triggered") before the malware is considered to be detected – we will present results when the trigger level *r* equals 2 or 3.

Of course many other voting configurations are possible, but we chose these two above as we believe they best represent the contrasting tradeoffs between detection rates and false alarm rates that decision makers and administrators should take into consideration when deciding on a system configuration. We must make clear to the reader that since we are dealing with confirmed malware samples we cannot really present any data about false positives. But the rooN results will allow us to get at least initial estimates of how the *sensitivity* (i.e. the rate of correct detections of confirmed malware) of the AV detection capability will be affected when an administrator wishes to use *majority voting* or *trigger-level* detection setup to attempt curtailing the false positive rate.

4.1 1-out-of-N Results

Table 4 gives the aggregated results for all the possible distinct 1ooN systems that can be built from these 26 AV products.

The total number of systems in each configuration (the penultimate, right-most column of Table 4) is obtained using the combinatorial formula ${}^{n}C_{r}$. For example, the total number of 1-out-of-2 (1oo2) systems that you can create from 26 different AVs is ${}^{26}C_{2}$ = 325, and so on. For each of these distinct systems we calculate the average failure rate for all instances of malware sent to them. The table only goes as far as 1-out-of-15 because we had no single demand that caused any combination of 15 different AVs to fail simultaneously. Hence from 1oo15 onwards we observe perfect detection with our dataset.

The first column of the results (f.r. = 0) shows the number of systems of each 1ooN configuration that detected all the malware on all the dates that they inspected them. We had no single AV that was in this category (as evidenced in Table 3), but this number grows substantially as the number of AVs in a diverse configuration increases (you can see its proportional growth to all the systems in a particular configuration in the right-most column of Table 4). The second column (1.E-05 ≤ f. r. < 1.E-04) shows the number of systems whose average failure rate for all instances of malware is between 10^{-4} and

TABLE 4. COUNTS OF 1-OUT-OF-N SYSTEMS OF AVS PER FAILURE RATE BAND. ABBREVIATION: F.R. - FAILURE RATE

Bands	*f. r. = 0*	*1.E-05 ≤ f. r.< 1.E-04*	*1.E-04 ≤ f. r.< 1.E-03*	*1.E-03 ≤ f. r.< 1.E-02*	*Total number of systems*	*Proportion of systems with perfect detection rate*
1oo2	80	40	74	131	325	0.246154
1oo3	1,353	395	528	324	2,600	0.520385
1oo4	10,985	1,618	1,881	466	14,950	0.734783
1oo5	57,033	4,169	4,134	444	65,780	0.867026
1oo6	216,199	7,355	6,360	316	230,230	0.939057
1oo7	641,030	9,255	7,355	160	657,800	0.974506
1oo8	1,547,183	8,522	6,516	54	1,562,275	0.990340
1oo9	3,114,327	5,827	4,385	11	3,124,550	0.996728
1oo10	5,306,587	2,952	2,195	1	5,311,735	0.999031
1oo11	7,724,287	1,082	791	0	7,726,160	0.999758
1oo12	9,657,234	272	194	0	9,657,700	0.999952
1oo13	10,400,529	42	29	0	10,400,600	0.999993
1oo14	9,657,695	3	2	0	9,657,700	0.999999
1oo15	7,726,160	0	0	0	7,726,160	1

10^{-5}. The worst case failure rate band for any diverse setup is 10^{-2} and 10^{-3}. In Table 3 we saw that the worst case failure rate band, even after filtering away the 6 worst single AVs, for any single AV is 10^{-1} and 10^{-2}. Hence, for this dataset, we can interpret this result as "*the worst diverse 1ooN configuration brings an order of magnitude improvement over the worst single AV product*".

Figure 4 shows the cumulative distribution function (cdf) of the failure rate achieved for single and multiple *1ooN* AV configurations. To get a more accurate cdf we drew the cdf curve over 100 failure rate bands between 0-1 rather than just 6 as we have shown in Table 4. We can clearly observe the shift of AV performance towards a lower failure rate when adding extra layers of diversity (i.e. additional diverse AVs).

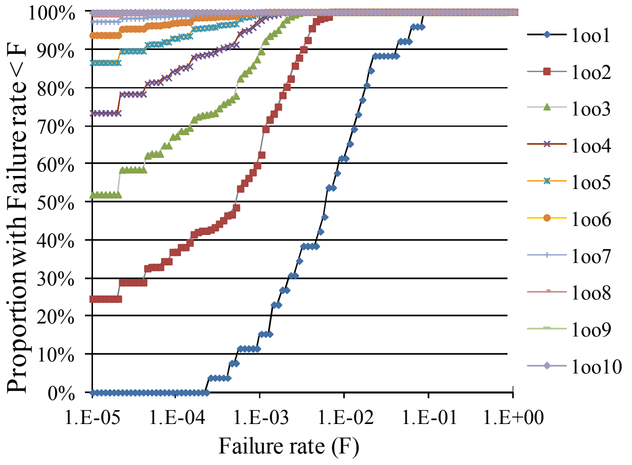


Fig. 4. Cumulative distribution of failure rate for different 1ooN configurations (up to 1oo10).

4.2 Majority Voting Results

Table 5 gives the results for all the possible distinct rooN majority voting systems that can be built from these 26 AV products. As mentioned previously, in this section we are interested in setups that allow majority voting, so we chose configurations where N is an odd number and r is equal to $(N+1)/2$. The structure of the table is similar to that of Table 4, apart from an added column 1.E-02 ≤ f. r. < 1.E-01 which shows the number of systems whose average failure rate for all instances of malware is between 10^{-2} and 10^{-1} (we had no 1ooN systems for any configuration in this failure rate band).

Comparing these results with those of Table 4, we see that the proportion of systems with perfect failure rates in each *rooN* configurations remains relatively low (no more than 5.45% in each configuration), and that the majority of systems for each configuration have a failure rate in the bands 1.E-04 ≤ f. r. < 1.E-02. We can observe this even more clearly by looking at Figures 5 and 6 which show the cumulative distribution function (cdf) of the failure rate achieved for *rooN* AV configurations. We have split these into two figures to make clearer the relative decrease in detection rates for majority voting configurations with high degree of diversity (from 8oo15 onwards).

We observe this decrease in detection rates, because for higher degree of diversity we require more AVs to detect a malware before we raise an alarm, which can deteriorate the detection rate. The main conclusion from this analysis is that the performance of diverse majority voting systems is not improved by adding further AVs.

Table 5. Counts of different r-out-of-N systems of AVs per failure rate band. Abbreviation: f.r. - failure rate

Bands	*f. r. = 0*	*1.E-05 ≤ f. r.< 1.E-04*	*1.E-04 ≤ f. r.< 1.E-03*	*1.E-03 ≤ f. r.< 1.E-02*	*1.E-02 ≤ f. r.< 1.E-01*	*Total number of systems*	*Proportion of perfect detection systems*
2oo3	119	122	453	1,891	15	2,600	0.045769
3oo5	3,224	4,022	16,548	41,965	21	65,780	0.049012
4oo7	34,462	42,685	204,795	375,850	8	657,800	0.052390
5oo9	168,386	198,079	1,128,905	1,629,180	0	3,124,550	0.053891
6oo11	421,073	459,406	3,125,369	3,720,312	0	7,726,160	0.054500
7oo13	563,168	560,985	4,608,873	4,667,574	0	10,400,600	0.054147
8oo15	407,204	364,129	3,690,040	3,264,787	0	7,726,160	0.052705
9oo17	156,518	121,195	1,586,216	1,260,621	0	3,124,550	0.050093
10oo19	30,117	18,941	350,383	258,359	0	657,800	0.045784
11oo21	2,520	1,149	36,094	26,017	0	65,780	0.038310
12oo23	65	15	1,434	1,086	0	2,600	0.025
13oo25	0	0	14	12	0	26	0

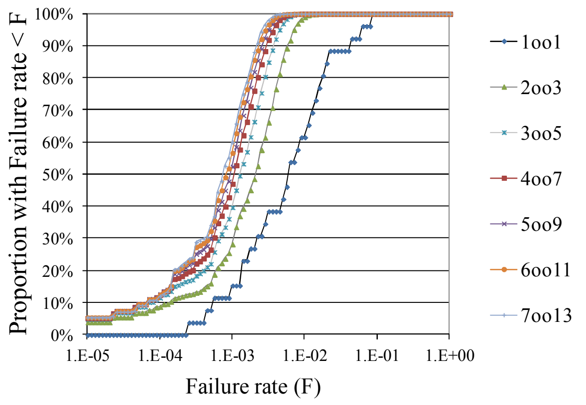


Fig. 5. Cumulative distribution of failure rate for different rooN configurations (majority voting up to 7oo13).

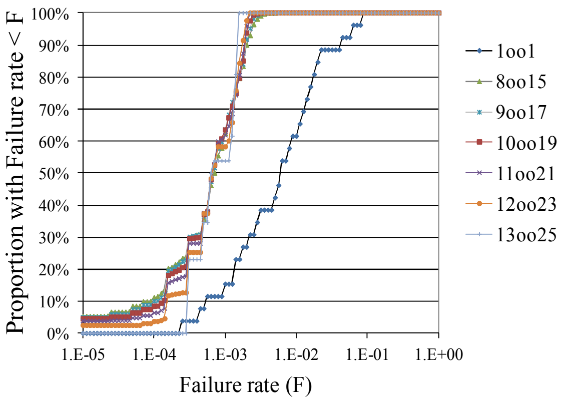


Fig. 6. Cumulative distribution of failure rate for different rooN configurations (majority voting from 8oo13 to 13oo25).

4.3 Trigger Level Results

An intermediate voting scheme between 1ooN and a majority vote is a trigger-level detection where some fixed number r out of N AVs have to agree to detect a malware.

This is a useful scheme in cases where the administrators wish to curtail the false positive rate by only raising an alarm (i.e. classifying a file as malicious) if at least some minimum number of AVs agrees that the file is malicious, but not necessarily go with a majority voting setup which as we saw previously can severely deteriorate the detection rates for confirmed malware (true positives).

We looked at the diverse setups with trigger levels of 2 and 3 (i.e. 2ooN and 3ooN setups) for N values up to 26.

The Figures 7 and 8 below show the *cdf*s for 2ooN and 3ooN respectively.

The conclusion drawn from this analysis is that detection with a fixed trigger level *r* is improved by increasing the number of diverse AVs (*N*).

Comparing and contrasting figures 4, 5, 6, 7 and 8 we can see that the rate of detection of malware for lower values of N is worse with rooN and qooN setups compared with 1ooN, but the deterioration is far less pronounced for qooN setups, especially for high values of N.

It is obvious that rooN and qooN configurations will have a lower detection rate than 1ooN ones. The question is how much lower. As we explained before, since we only have confirmed malware samples we cannot measure the false positive rate. However, the qooN and rooN configurations allow us to measure the sensitivity (the

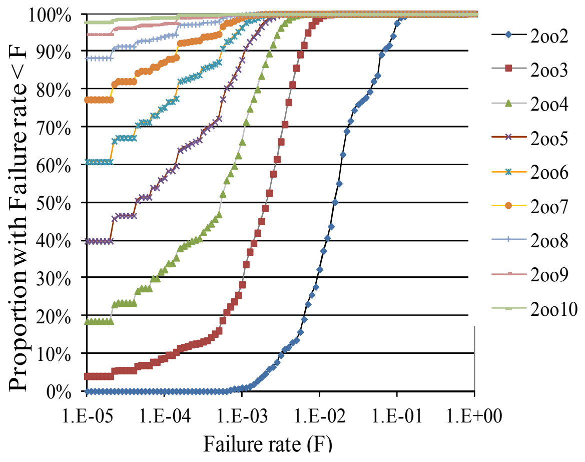


Fig. 7. Cumulative distribution of failure rate for different 2ooN configurations (up to 2oo10).

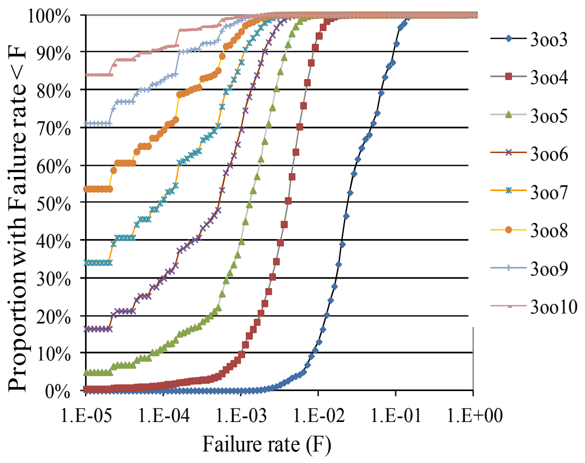


Fig. 8. Cumulative distribution of failure rate for different 3ooN configurations (up to 3oo10).

false negative rate) – the rate of missed detections for confirmed malware – that would result from a decision by administrators to keep down the false positive rate. We can do this by comparing the 1ooN results with qooN and rooN results for a given N.

4.4 Comparison of the results of the various diverse setups

Figures 9, 10 and 11 contain cumulative distribution of ratios of failure rates for rooN, 2ooN and 3ooN respectively over 1ooN configurations, for a given N. With reference to Figure 9, we divided the proportion of systems that have a failure rate less than F for a given rooN configuration (i.e. the points in the lines of a given configuration in Figures 5 and 6) with the corresponding proportion, for the same N, of systems that have a failure rate less than F for a given 1ooN configuration (i.e. the points in lines of a given configuration in Figures 4).

For example, for 2oo3/1oo3 in Figure 9: to calculate the first point in the line for this configuration we divide the proportion of 2oo3 systems that have a failure rate less than 1E-05 (from Table 5 there are 119/2,600 systems with perfect detection) with the corresponding proportion of 1oo3 systems that have a failure rate less than 1E-05 (from Table 4 there are 1,353/2,600 systems with perfect detection). This means there is more than an order of magnitude drop in the number of 3-version systems that have a perfect detection rate when we move from a 1oo3 to a 2oo3 setup. This drop in proportion of perfect systems when switching from a 1ooN to a rooN setup is even more pronounced for configurations with a higher N. The drop is much less pronounced for 2ooN and 3ooN setups which for higher values of N are all within an order of magnitude.

The figures confirm what we have observed in the previous sub-sections: clearly majority voting seems to be a very costly (on average) configuration in terms of the detection rate of confirmed malware, whereas 2ooN and 3ooN are less so. This cost may be compensated by a decrease in false positive rates, which unfortunately we cannot calculate with our dataset.

Another interesting comparison of the 1ooN and qooN setups is the proportion of systems within a given configuration N that have non-zero detection rates. Figure 12 presents this comparison.

We can see that to get 90% of all 1ooN systems to have a perfect detection of all malware with our dataset we need N~5 (we can observe this in Figure 12 by following where the x-axis value 5 meets the y-axis value 1.E-01 for the 1ooN line). To get 90% of 2ooN systems to detect all malware we need N~8 whereas N for 3ooN is > 10. This is another way in which an administrator could measure the cost of seeking confirmation from 2 or 3 AVs before raising alarms.

4.5 A Closer Look at the Simplest Diverse Configurations

We will now look more closely at the simplest diverse configurations, namely 1oo2 and 2oo3. We want to compare what improvements in malware detection they bring

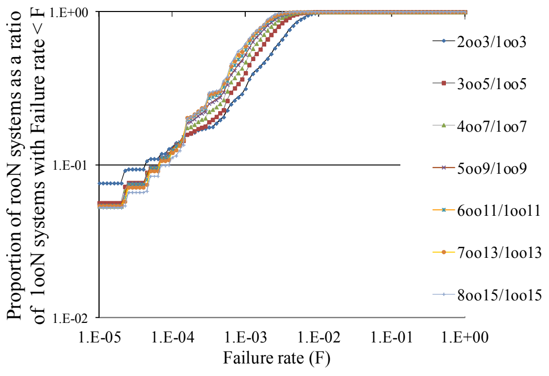


Fig. 9. Proportion of rooN systems as a ratio of 1ooN systems with Failure rate < F.

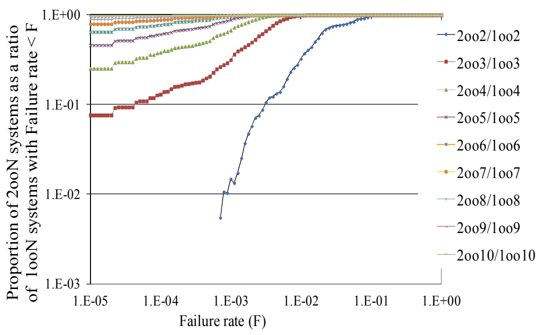


Fig. 10. Proportion of 2ooN systems as a ratio of 1ooN systems with Failure rate < F.

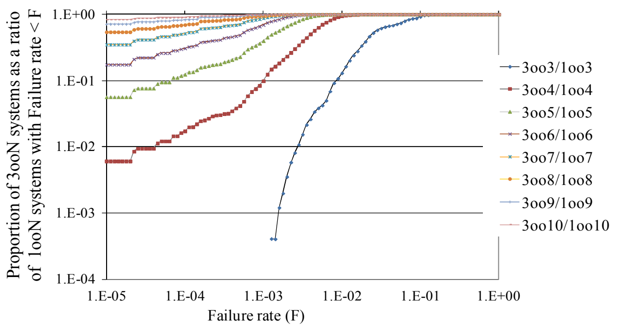


Fig. 11. Proportion of 3ooN systems as a ratio of 1ooN systems with Failure rate < F.

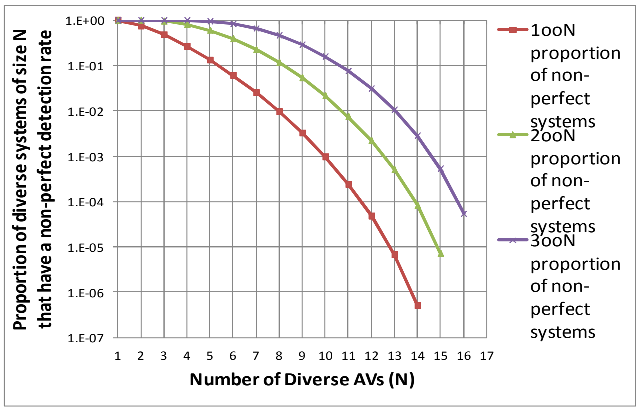


Fig. 12. Proportion of diverse systems of size N that have a non-perfect detection rate.

compared with single versions. The graph in Figure 13 shows the average improvements in the detection rate of

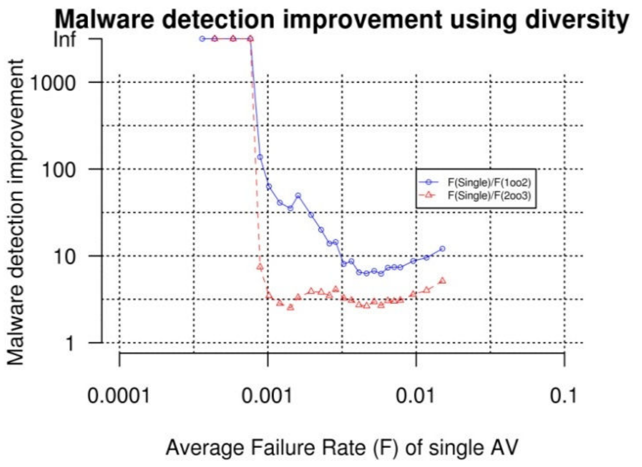


Fig. 13. Malware detection improvement over single versions using the simplest diverse configurations (1oo2 and 2oo3). Abbreviation: F- failure rate.

diverse systems over the average detection rates of single AVs.

The x-axis shows the average failure rates of a pool of single AVs as we remove the AV with the worst failure rate. So the right-most point is the average failure rate of all 26 AVs. We then progressively remove the AV with the worst detection rate, until we are left with the two AVs with best detection rates (we need at least two to build a 1oo2 system). The y-axis shows the ratio between the average failure rates of single version and 1oo2 configurations (shown with the blue line in the graph) and average failure rates of single version and 2oo3 configurations (shown with the red line in the graph).

We can see that removing the worst 11 single AVs the improvement rate for both 1oo2 and 2oo3 over the single versions remains fairly constant (around one order of magnitude for 1oo2 and around 5-fold for 2oo3). For the next 9 we see the improvements for 1oo2 getting larger (surpassing 2 orders of magnitude at some stage) though they remain constant for 2oo3. Finally for the last remaining ones, where the single AVs have high detection rates, the improvement ratio is then shown on the graph by the Infinity label (Inf) since the average failure rates of 1oo2 and 2oo3 systems are 0.

Figure 14 compares the average failure rates of 1oo2 and 2oo3 as we progressively remove the single worst AV. The blue line shows the ratio between the average failure rates of 2oo3 and 1oo2 configurations at each step (we discarded the last three steps where, as shown in figure 13, all 1oo2 and 2oo3 average failure rates were 0). We see that in some cases the average failure rate of a 1oo2 system becomes worse compared with the one from the previous step. This is because even though we have removed a single AV with the worst detection rate from the previous step that may not necessarily lead to the average failure rate of 1oo2 systems becoming smaller: what matters is how diverse the detection behavior of the AVs are.

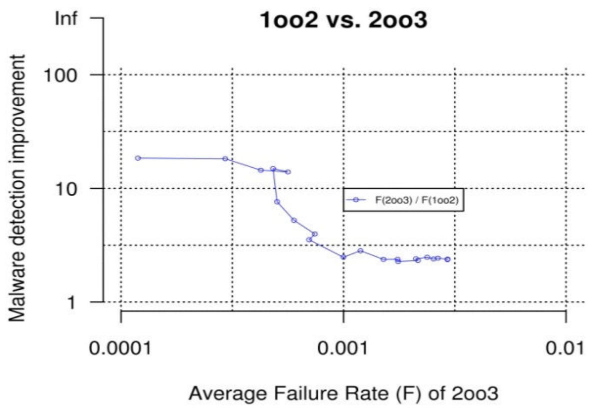


Fig. 14. Malware detection improvement of 1oo2 over 2oo3 configurations. Abbreviation: F- failure rate.

5 "Diversity over Time" Analysis

As we stated before, each malware sample was sent to a given AV product on several consecutive days after it was first uncovered in the SGNET honeypots. This period was a maximum of 30 days in our dataset. By their very nature signature-based AVs should eventually detect all the malware, provided the vendors write the appropriate signatures to detect them. However the response time for writing these signatures can differ significantly. Hence we analyzed to what extent the use of diverse AVs may help a system reduce its "at risk time", i.e. the time before a signature is available for a new malware. By using more than one AV, in a *1ooN* configuration for example, the system is only exposed for the time it takes the fastest of the vendors to define the signature.

Figure 15 shows a 3-dimensional plot of the entire dataset of the 26 AVs in our study.

We now look in more detail at the most "difficult" 10% of malware (i.e. the ones which were most often missed overall by the AV products). This is shown in Figure 16. Note that this not a zoom of the top-most 10 % of the Figure 15, because the ordering is done over the malware. The ordering is bottom-to-top, from the most difficult to detect to the least difficult (within this 10% of malware).

The x-axis lists the 26 AVs ordered from left to right based on the detection rate: from the highest detection rate to the one with the lowest. The y-axis contains a listing of all 1599 malware. The z-axis values are given by the intensity of the color on the plot, which is given by the legend on the right-hand side of the plot. The black colored lines are associated with malware that were never seen by a given AV (i.e. the missing data): in our dataset these are associated with just one AV product. The white lines are associated with malware which were always detected by given AV product. The values 1-30 in the z-axis, represented by colors ranging from intense green to light pink, are associated with the number of days that a given AV failed to detect a given malware in our study, but which eventually were detected. The red lines represent the malware which, in our collection period, were never detected by a given AV product.

The graph shows clear evidence of diversity, which is consistent with the results we have observed in the previous section, especially for *1ooN* configurations.

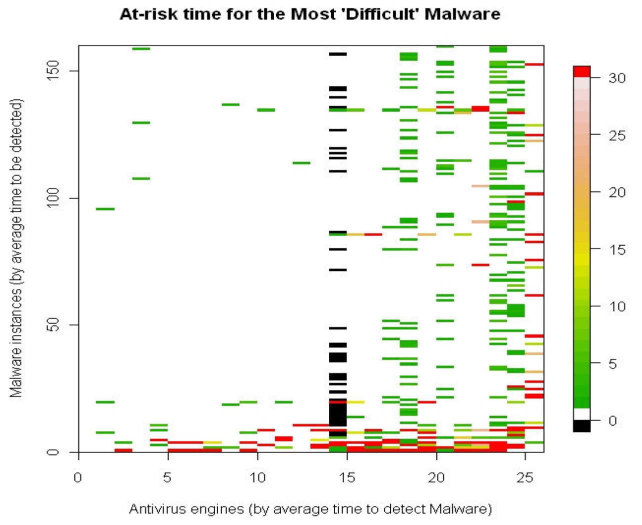


Fig. 16. The "at risk time" in number of days for each of the 26 AVs when considering the 160 most "difficult" Malware samples.

If we look at the graph along the x-axis it is clear again that there are infrequent cases of a line (i.e. a particular malware) running across the different AVs with the same color. This indicates that even for malware which, on average, different AVs are finding difficult to detect, there is considerable variation on which ones they find difficult (i.e. a malware which an AV finds difficult another one does not, and vice versa) and when they do fail, their at risk time does vary. Hence it is clear that using diverse AVs, even in the cases when different AVs fail to detect the malware, can have an impact in reducing the "at risk time" of a system: for example, in a 1ooN configuration, the time the system is at risk is the minimum time it takes any of the diverse AVs employed to detect the malware (.

Another viewpoint of the time dimension analysis is given in Figure 17. For each AV we looked at the time (in number of days) it took to detect a given malware sample. To make the graph more readable we categorized the malware into groups depending on the time it took an AV to detect them: always detected (shown on the bars filled

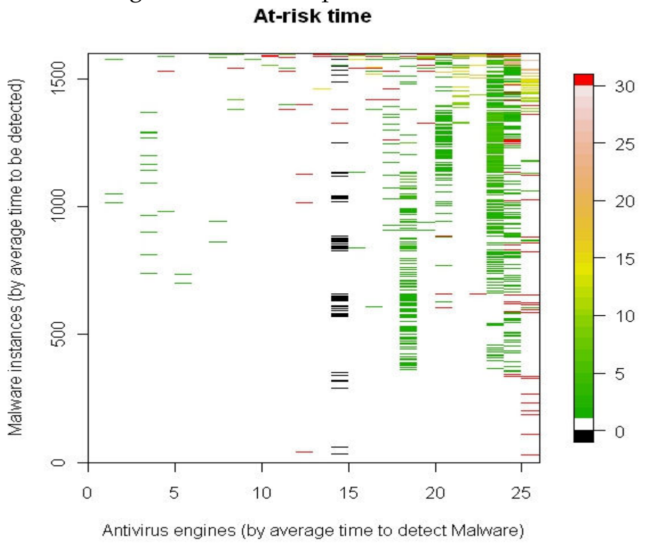


Fig. 15. The "at risk time" in number of days for each of the 26 AVs on each of the 1599 malware samples in our dataset.

in white color with black borders), never detected (shown in red fill color), no data (in black fill color), and then several 5-day groupings (1-5 days, 6-10 days etc.). Each bar shows the proportion of malware that are in any of these categories. We are only showing the top 45% in the y-axis, as all AVs always detected at least 55% of all the malware samples sent to them.

Figure 17 allows us to look more clearly at the empirical distribution of the time it takes each AV to detect the malware samples in our study. Along the x-axis the AVs are ordered left-to-right from the AV with the lowest at risk time (AV-7) to the highest (AV-5).

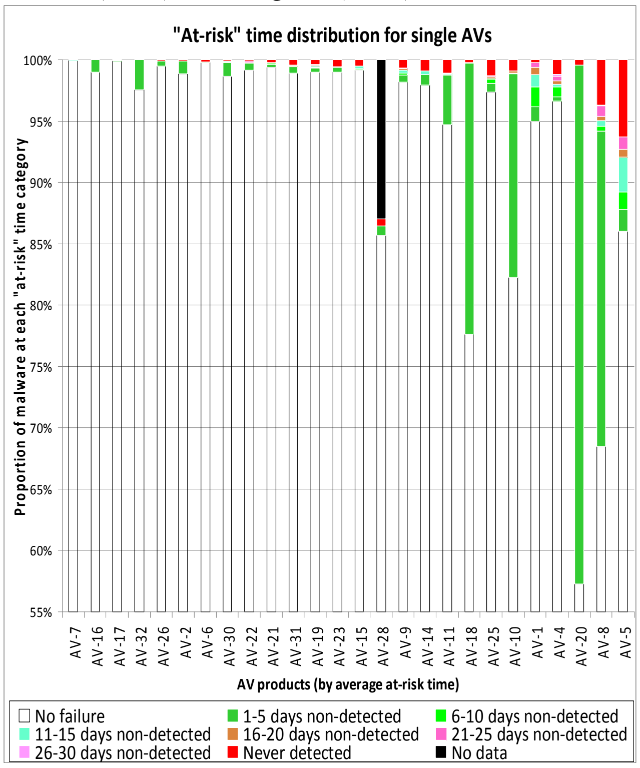


Fig. 17. "At risk time" distribution for single AVs.

The distribution of detection times may have a bearing on the choice of AVs that a given user may wish to make for a given system. Rather than choosing the "best on average", users may choose the AVs that do not have too many undetected malware (shown in red color in the bars of the graph in Figure 17). For example, even though AV-23 has a better (shorter) average "at risk time" than AV-18, it is evident from Figure 17 that AV-18 has a shorter red colored section of the bar compared with AV23. This means AV-18 failed to detect, in our study, a smaller number of malware than AV-23, even though its average at risk time was worse (higher).

From the viewpoint of diversity analysis, the choice of which set of AVs to choose is therefore not only influenced by the highest detection rates (diversity in "space") but also the time it takes the different AVs to detect malware (diversity in "time"). Different AV vendors may have different policies on deciding which malware they prioritize for a signature definition. Therefore the needs of an individual end-user for whom a definition of a signature for a particular malware sample is of high importance, may mismatch with the AV vendor's prioritization of that malware. So use of diverse AVs helps avoid this "vendor lock-in". Selection of diverse AVs to optimize the diversity in time aspect requires we choose AVs from vendors that exhibit diverse policies at prioritizing signature definitions for different malware.

6 Modeling the Impact of Additional AVs

6.1 Exponential Power Law Models for Probability of Imperfect 1ooN, 2ooN and 3ooN Systems

An interesting characteristic of the data plotted in Figure 12 is the asymptotic reduction in the proportion of "imperfect" (i.e. systems that fail to detect all the malware) 1ooN, 2ooN and 3ooN systems. When reasoning about 1ooN, then a simple exponential model of the decrease with the number of AVs (N) in the diverse system would estimate the proportion as:

$$P_{faulty}(1ooN) = p^N \qquad (1)$$

where p is the probability that a single AV fails to detect all the malware.

In practice this proved to be a very poor fit to the observed proportions. However an empirically derived exponential power law model of the form:

$$P_{faulty}(1ooN) = p^{N^2} \qquad (2)$$

proved to be a good fit to the observed data for 1ooN as shown in Figure 18. The goodness of fit was assessed by performing a linear regression analysis of the linearised form of the relationship, i.e. $\ln(P(faulty_{1ooN})) \propto N^2$. The coefficient of determination, R^2 was extremely high at 0.9975.

Figure 18 also shows the fit obtained using an exponential power law model fit for the 2ooN and 3ooN results. We obtained the best fits using the equations:

$$P_{faulty}(2ooN) = a^{(N-1)^{2.5}} \qquad (3)$$

$$P_{\text{faulty}}(3ooN) = b^{(N-2)^3} \qquad (4)$$

Again, there is a very good fit when regression is applied to linearised versions of these relationships (R^2 values of 0.9985 and 0.9990 respectively).

We have also compared the 2ooN and 3ooN cases against a "shifted" 1ooN distribution, where we essentially take the fitted 1ooN curve and shift it to the right. Effectively we assume:

$$P_{faulty}(2ooN) = p^{(N-2)^2} \qquad (5)$$

and

$$P_{faulty}(3ooN) = p^{(N-4)^2} \qquad (6)$$

As we can see in Figure 18, the curves are a reasonable fit to the empirical data (with an R^2 value better than 0.97) although it is not as good as the fit obtained using a higher power law. This result suggests there could be a useful "rule of thumb" for implementing a diverse AV voting scheme. If you increase the trigger level by one you need to increase the total number of AVs – N - by two to get the same probability of selecting a perfect combination of AVs for a given detection trigger level.

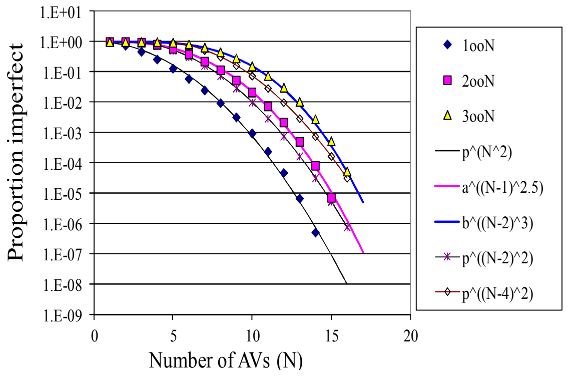


Fig. 18. Effect of increasing diversity by adding AVs.

6.2 Further Analysis of the Modeling for 1ooN

In our earlier paper [12] we had no detailed explanation for this surprisingly good fit of the 1ooN system detection capability to the exponential power law model. We have looked into this aspect in more detail for the 1ooN setup.

One way of exploring the data is to consider what happens to the initial probability distribution of failure rates as we combine "barriers" (i.e. different AVs). In this model we take the barriers to be a composite one with a probability distribution given by combining the rates of the different AVs. We can then model how the distribution changes as we add barriers.

As we add independent barriers we are making a convolution of the distribution with itself. To see how the shape changes we could first take a Laplace transform then multiply the resulting distribution in the Laplace transform space. The reverse transformation would give us the shape of the new distribution we can then adjust for normalization etc. Although this is tractable for Gamma distributions, the data we have did not provide a convincing fit. Instead we have qualitative arguments, due to the multiplicative nature of the barriers, that the distributions might tend towards log normality. For these there is no straightforward transform.

However we can use the extreme value result due to [18] that the probability density function (*pdf*) of the product of n independent identically distributed positive random variables (the probability of failure of the AVs) is:

$$AV_n = \prod_n av_i \tag{7}$$

with individual pdfs p(x), under certain assumptions, given by the expression from [18]:

$$P_n(X) \sim \left[p(X^{1/n})\right]^n \tag{8}$$

for $X \to \infty$ and n finite.

To use this approximation we need a suitable property that satisfies the condition $X \to \infty$, such as the time to failure (in terms of number of demands) of the AVs. So the *time to failure* (*ttf*) distribution for n barriers can be derived from the individual distributions and (8). Note that if we are interested in probability of failure after D demands we see the most relevant part of the distribution for m barriers is only between D/n and D.

We can use equation (8) to see the result of multiplying independent variables that are distributed log normally:

$$p(x) = \frac{A}{\sigma x}\exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right) \tag{9}$$

Applying (8), we get an analytical result for the scaling with the number of barriers N that is approximately given by

$$P(N,x) = \left[A_N \frac{1}{x^{1/N}} exp\left(-\frac{(ln(x)-\mu*N)^2}{N^2\, 2\sigma^2}\right)\right]^N \tag{10}$$

which simplifies to the log-normal distribution with parameters:

$$\mu_n = N.\mu_0 \quad \text{and} \quad \sigma_n = \sqrt{N}.\sigma_0 \tag{11}$$

With the log-normal mean given by

$$\exp\left(\mu_n + \frac{{\sigma_n}^2}{2}\right) \tag{12}$$

The empirical results of Figure 19 show the probability of failure of the AVs after a total of D demands (ignoring for now the differences between the demands the AVs see). This provides a measure of the probability, or confidence, that the failure rate will be above ~1/D$_t$. We can calculate how the probability that the failure rate will be above a threshold changes, i.e. our confidence in the rate being greater than, say, 0.0001. This incomplete log normal function has been calculated and the variation with N given from (10).

The variation in the incomplete cumulative distribution with the number of barriers is shown in Figure 19. The dashed lines are the fit with different lower bounds on the failure rate. The actual data is shown in red points and the original exponential power law fit is the dotted red line.

Note that the log-normal has a power law tail over most of this range, independent of N, and hence does not curve like the exponential power law fit.

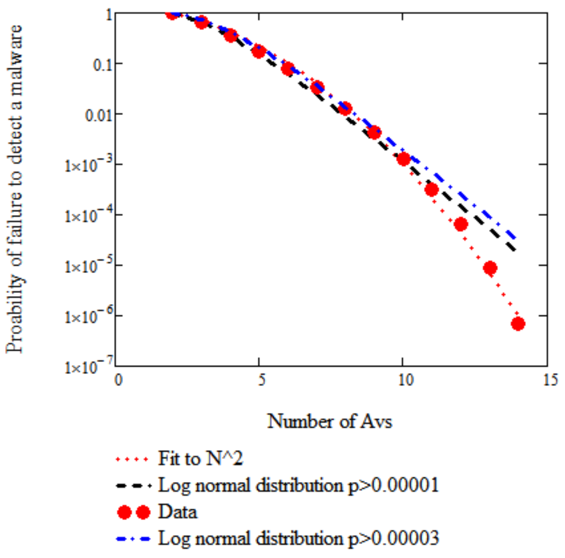


Fig. 19. Variation in the incomplete cumulative distribution with the number of barriers.

7 Discussion

7.1 Confidence in the Best Case Observed Failure Rate

Any claim about the best case (i.e. lowest) failure rate we can hope for, using this dataset, will be in the region of 10^{-4} to 10^{-5}. This is because we have a maximum of 47,970 demands sent to any given AV (as we explained earlier). This is the upper bound on the observed failure rate. To make claims about lower failure rates (i.e. smaller than 1/47,970) we need more data – or, in other words, we must do more testing. The theoretical background behind these intuitive results is given in [19][3].

But the fact that we are seeing only, for instance, 5 out of almost 10 million different combinations of 1oo14 systems fail on this dataset is telling us something about the improvements in detection rates from using diversity in a 1ooN setup (cf. Table 4), even if we cannot claim a failure rate lower than 1/47,970. We may interpret the 0.999999 proportion of 1oo14 systems with a perfect failure rate as *a 99.9999% confidence we have in the claim that, for this dataset, "the average failure rate of a 1oo14 diverse AV system constructed from randomly selecting 14 AVs from a pool of 26 AV products, will be no worse than ~ $(1 / 4.8) * 10^{-5}$"*. This may or may not persuade a given system administrator to take on the additional cost of buying this extra security: this will clearly depend on their particular requirements. But in some cases, especially regulated industries, there may be a legal requirement to show actually achieved failure rates with a given level of confidence. It may not be feasible to demonstrate those failure rates with the associated confidence for single AV products without extensive testing.

The claim above is limited by the level of diversity we can expect between various diverse configurations. We "only" have 26 AV products (even though this may represent a large chunk of the available commercial solutions available). Hence, for instance, even though we have over 65,000 different combinations of 1oo5 systems using 26 AVs, the level of diversity that exists between these different 1oo5 systems may be limited in some cases. For example, a 1oo5 system which consists of AV1, AV2, AV3, AV4 and AV5, is not that much different from a 1oo5 system consisting of AV1, AV2, AV3, AV4 and AV6. Hence care must be taken from generalizing the level of confidence from these large numbers without considerations of these subtle concepts on the sample space of possible systems.

7.2 Limitations of the Study and the Dataset

There are two major limitations of the dataset we have used which prevent us from making more generalized conclusions on the possible benefits of diversity.

The first is to do with the time in which the data has been collected. As we mentioned before the malware samples in our study were collected from the SGNET network in 2008, hence these malware samples do not accurately reflect the current prevalence of malware in 2012. However we should stress that we are matching *like* with *like*: i.e. malware samples and AVs at some snapshot in time, in this case 2008. We plan to do further work with malware which are prevalent in 2012 and the detection capabilities of 2012 AVs when they inspect these malware. This will give us results on the consistency, or not, of the detection capabilities of diverse AVs in different snapshots in time.

The second limitation is due to the nature of the data set: we only have confirmed malware. Hence we can measure failure to detect genuine malware (*false negatives*), but we could not measure cases where benign files are incorrectly identified as malware (*false positives*). False positive rate is an important measure when evaluating the effectiveness of any detection system, including AV systems. Of course we could have artificially created a large set of non-malicious files which we could have sent to the AV systems and obtained false positive rates this way. But this can just as easily be criticized due to the lack of representativeness of the chosen non-malicious files. Choosing representative test loads and defining what constitutes a false positive is a matter of some debate in the AV community (see discussion in [11, 14] for more details). We plan to study the false positive aspects in future work.

7.3 Discussion of the Exponential Power Law Model and its Implications on Systems with Diverse Barriers

There is no immediate explanation for the good fit of the 1ooN system detection capability to the exponential power law model. Although the results in Section 6.2. show how single barrier *pfd* can lead to a distribution parameterized on the number of AV barriers, *N*, an analytical derivation of the N^2 empirical result has eluded us. It would seem that the initial fit over the first 10 barriers can be modeled by a combination of independent random variables, but not the complete set of empirically derived points. We have undertaken a number of simulations which suggest that the underlying variability in AV failure rates is more important than their lack of independence, but dependency might be more significant for the tail of the distribution. We are continuing with these investigations.

If the exponential power distribution is shown to apply on other datasets, it would be a useful means for predicting the expected detection rates for a large 1ooN AV system based on measurements made with simpler 1oo2 or 1oo3 configurations. Also, quite significantly, the results show that asking the question "*how does the chance of surviving D demands vary with N barriers*" provides a result that is much better, in the sense of a much lower probability of failure than one might expect from a straightforward consideration and even from an optimistic independence assumption of the means.

8 Related Work

8.1 Architectures that Utilize Diverse AntiVirus Products

In this paper we have presented the potential gains in detection capability that can be achieved by using diverse AVs. Security professionals may be interested in the architectures that enable the use of diverse AV products.

An initial implementation of an architecture called Cloud-AV has been provided in [4], which utilizes multi-

[3] Section 3 of [19] contains a discussion of the limits on claims we can make about reliability when statistical testing reveals no failures.

ple diverse AntiVirus products. The Cloud-AV architecture is based on the client-server paradigm. Each host machine in a network runs a host service which monitors the host and forwards suspicious files to a centralized network service. This centralized service uses a set of diverse AntiVirus products to examine the file, and based on the adopted security policy makes a decision regarding maliciousness of the file. This decision is then forwarded to the host. To improve performance, the host service adds the decision to its local repository of previously analyzed files. Hence, subsequent encounters of the same file by the host will be decided locally. The implementation detailed in [4] handles executable files only. A study with a deployment of the Cloud-AV implementation in a university network over a six month period is given in [4]. For the executable files observed in the study, the network overhead and the time needed for an AntiVirus engine to make a decision are relatively low. This is because the processes running on the local host, during the observation period, could make a decision on the maliciousness of the file in more than 99% of the cases that they had to examine a file. The authors acknowledge that the performance penalties might be significantly higher if the types of files that are examined increases as well as if the number of new files that are observed on the host is high (since the host will need to forward the files for examination to the network service more often).

Commercial solutions that use diverse AV engines for file and for e-mail scanning are available from [5-7].

8.2 Other Related Empirical Work with AV Products

Empirical analyses of the benefits of diversity with diverse AV products are scarce. CloudAV [4] and our previous papers [12], [13] are the only published research we know of.

Studies which perform analysis of the detection capabilities and rank various AV products are, on the other hand, much more common. A recent publication [9] reports results on ranking of several AV products and also contains an interesting analysis of "at risk time" for single AV products. Several other sites[4] also report rankings and comparisons of AV products, though care must be taken when comparing the results from different reports, as they might use different definitions of "system under test".

9 Conclusions

We presented results of an assessment of the application of diversity in a real world scenario based on realistic data generated by a distributed honeypot deployment.

The performance analysis of the signature-based components showed a considerable variability in their ability to correctly detect the 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 diverse performance of the detectors justified the exploitation of diversity to improve the detection performance. We calculated the failure rates of all the possible diverse systems in various 1-out-of-N, trigger-level (where a minimum number of 2 or 3 AV detections is needed before an alarm is raised) and majority voting setups (where we need a majority of AVs in a setup to detect a malware before an alarm is raised).

As shown in [14], the comprehensive performance evaluation 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 main results can be summarized as follows:

- Despite the generally high detection rates of the AVs, none of them achieved 100% detection rate;
- The detection failures were both due to an incomplete signature databases at the time in which the samples were first submitted for inspection, but also due to regressions in the ability to detect previously known samples as a consequence, possibly, of the deletion of some signatures;
- Considerable improvements in detection rates can be gained from employing diverse AVs;
  - No single AV product detected all the malware in our study, but almost 25% of all the diverse pairs, and over 50% of all triplets in 1-out-of-N configurations successfully detected all the malware;
  - In our dataset, no malware causes more than 14 different AVs to fail on any given date. Hence we get perfect detection rates, with this dataset, by using 15 diverse AVs in a 1-out-of-N configuration;
  - The detection rates are lower for "trigger level detection" (2-out-of-N and 3-out-of-N) and majority voting (r-out-of-N) setups compared with 1-out-of-N but on average are better than using a single AV;
- Significant potential gains in reducing the "at risk time" of a system from employing diverse AVs: even in cases where AVs fail to detect a malware, there is diversity in the time it takes different vendors to successfully define a signature to detect a malware;
  - The analysis of "at risk time" is a novel contribution compared with traditional analysis of benefits of diversity for reliability: analysis and modeling of diversity has usually been in terms of demands (i.e. in space) without considering the time dimension;
- An empirically derived exponential power law model proved to be a good fit to the proportion of systems in each simple detection (1-out-of-N) and trigger level detection (2-out-of-N and 3-out-of-N) diverse setup that had a zero failure rate. If this was found to be generic for other datasets, it would be a useful and cost-effective means for predicting the probability of perfect detection for systems that use a large number of AVs (or other types of detection systems) based on measurements made with systems that are composed of fewer (say 2 or 3) AVs.
- A composite barrier model partially explains the observed behavior, but it requires more development. If the impact of adding barriers can be accurately (or

[4] av-comparatives.org/, av-test.org/, virusbtn.com/index

even pessimistically) modeled, it could have significant implications for defense in depth and the architecture of safety and security critical systems.

10 Further Work

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 [14] 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 analyzing the behavioral characteristics of the inspected samples, it is very difficult to setup a benchmark capable 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, with other more up to date datasets, to test the detection capabilities of these products in full, including their sensitivities to false positives (whatever the definition of a false positive may be for a given setup).

Other provisions for further work include:

- 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;
- Studying the detection capabilities with datasets that allow measurements of false positives in addition to false negative rates. This will allow us a better analysis of the tradeoffs between the various 1-out-of-N, trigger-level and majority voting detection setups;
- Though we presented various results and distributions that show varying efficacy of different types of diversity, in practice a user will have to commit to a particular diverse configuration. We plan further work to study this in more detail. For example, we plan to study whether the particular combinations of AVs that are best for 1-out-of-N are the same ones that are best for other configurations? Are these AVs generally the ones that are individually best? Etc.
- More extensive exploratory modeling and modeling for prediction. The results with the exponential power law distribution were a pleasant surprise, but more extensive analysis with new datasets is required to make more general conclusions. More theoretical work is needed to relate the combined asymptotic results to the underlying distributions. Extensions of current methods for modeling diversity to incorporate the time dimension are also needed.

Acknowledgment

We would like to thank our colleagues at CSR for fruitful discussions of the research presented, and especially Bev Littlewood, Andrey Povyakalo and Robert Stroud who reviewed an earlier version of this paper.

References

[1] van der Meulen, M.J.P., S. Riddle, L. Strigini and N. Jefferson. *Protective Wrapping of Off-the-Shelf Components*. in *the 4th Int. Conf. on COTS-Based Software Systems (ICCBSS)*. Bilbao, Spain, p. 168-177, 2005.

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

[3] Reynolds, J., J. Just, E. Lawson, L. Clough, R. Maglich, and K. Levitt. *The Design and Implementation of an Intrusion Tolerant System*. in *32nd IEEE Int. Conf. on Dependable Systems and Networks (DSN)*. Washington, D.C., USA, p. 285-292, 2002.

[4] Oberheide, J., E. Cooke and F. Jahanian. *Cloudav: N-Version Antivirus in the Network Cloud*. in *Proc. of the 17th USENIX Security Symposium*, p. 91–106, 2008.

[5] GFi. *Gfimaildefence Suite*, last checked 2012, http://www.gfi.com/maildefense/.

[6] Microsoft. *Microsoft Forefront Protection*, last checked 2012, http://sharepoint.microsoft.com/en-us/product/Related-Technologies/Pages/Forefront-Protection-2010-for-SharePoint.aspx.

[7] GData. *Gdata Internet Security*, last checked 2012, http://www.pcworld.com/article/165600/gdata_internet_security_2010.html.

[8] VirusTotal. *Virustotal - a Service for Analysing Suspicious Files*, last checked 2012, http://www.virustotal.com/sobre.html.

[9] Sukwong, O., H.S. Kim and J.C. Hoe, *Commercial Antivirus Software Effectiveness: An Empirical Study.* IEEE Computer, 44(3): p. 63-70. 2011.

[10] Leita, C. and M. Dacier. *Sgnet: Implementation Insights*. in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. Salvador da Bahia, Brazil, p. 1075-1078, 2008.

[11] Leita, C., M. Dacier, J. Canto and E. Kirda. *Large Scale Malware Collection: Lessons Learned*. in *Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems, 27th Int. Symp. on Reliable Distributed Systems (SRDS)*. Napoli, Italy, 2008.

[12] Bishop, P., R. Bloomfield, I. Gashi and V. Stankovic. *Diversity for Security: A Study with Off-the-Shelf Antivirus Engines*. in *Software Reliability Engineering (ISSRE), 2011 IEEE 22nd International Symposium on*, p. 11-19, 2011.

[13] Gashi, I., C. Leita, O. Thonnard and V. Stankovic. *An Experimental Study of Diversity with Off-the-Shelf Antivirus Engines*. in *Proceedings of the 8th IEEE Int. Symp. on Network Computing and Applications (NCA)*, p. 4-11, 2009.

[14] Leita, C. *Sgnet: Automated Protocol Learning for the Observation of Malicious Threats*, University of Nice, Nice, France, 2008.

[15] Bayer, U. *Ttanalyze: A Tool for Analyzing Malware*, Technical University of Vienna, Vienna, Austria. p. 86, 2005.

[16] Bayer, U., A. Moser, C. Kruegel and E. Kirda, *Dynamic Analysis of Malicious Code.* Journal in Computer Virology, 2(1): p. 67–77. 2006.

[17] F-Secure. *Malware Information Pages: Allaple.A*, last checked 2011, http://www.f-secure.com/v-descs/allaple_a.shtml.

[18] Frisch, U. and D. Sornette, *Extreme Deviations and Applications.* J. Phys. I France, **7**(9): p. 1155-1171. 1997.

[19] Littlewood, B. and L. Strigini, *Validation of Ultrahigh Dependability for Software-Based Systems.* Commun. ACM, 36(11): p. 69-80. 1993.

[20] Thomas, M. and R.D. Reiss, *Statistical Analysis of Extreme Values: With Applications to Insurance, Finance, Hydrology and Other Fields*. 2nd ed, Birkhauser. 2001