A Study of the Relationship Between Antivirus Regressions and Label Changes

Ilir Gashi[1], Bertrand Sobesto[2], Stephen Mason[1], Vladimir Stankovic[1], and Michel Cukier[2]
[1] Centre for Software Reliability, City University London, London, UK
[2] University of Maryland, College Park, MD, USA
[1]{ilir.gashi.1, stephen.mason.1, vladimir.stankovic.1}@city.ac.uk, [2]{mcukier, bsobesto}@umd.edu

***Abstract* — AntiVirus (AV) products use multiple components to detect malware. A component which is found in virtually all AVs is the signature-based detection engine: this component assigns a particular signature label to a malware that the AV detects. In previous analysis [1-3], we observed cases of *regressions* in several different AVs: i.e. cases where on a particular date a given AV detects a given malware but on a later date the same AV fails to detect the same malware. We studied this aspect further by analyzing the only externally observable behaviors from these AVs, namely whether AV engines detect a malware and what labels they assign to the detected malware. In this paper we present the results of the analysis about the relationship between the changing of the labels with which AV vendors recognize malware and the AV regressions.**

Keywords - security assessment; empirical analysis; malware; antivirus; intrusion detection.

I. INTRODUCTION

AntiVirus (AV) products are one of the most commonly deployed security protection software in both personal and business deployments. Almost all computers currently connected to the Internet have some type of an AV product installed. Modern AV products use many different components to improve their detection capabilities, ranging from signature-based detection engines to anomaly based statistical patterns and reputation based data collection services.

Studies which perform analysis of malware detection capabilities and rank various AV products are very common. Several sites [1] 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".

Our own previous research has concentrated on the benefits that diversity brings in improving the detection capabilities of the AV products i.e. using more than one AV product, from different vendors, in a diverse configuration to improve the malware detection capability [1-3]. While performing this research we observed an interesting phenomenon: AV products seem to *regress* in their detection capability, i.e. they failed to detect a malware which they had detected successfully in the past. We also noticed that many AV products at different times would assign *different labels* when identifying the same piece of malware.

We speculate that AV vendors are under great competitive pressure to update their rulesets frequently and be able to detect malware with as small ruleset as possible (due to the need to not only achieve high detection rates but also to do this in short response times and without consuming too many computing resources on the host machines). The frequent changes of the rulesets (which we observe externally as the labels with which the AV products identify the malware) may cause regressions. To check whether there is enough empirical evidence to support this claim we decided to run empirical studies.

Our experimental infrastructure consists of a honeypot network distributed over several different countries. These honeypots collect malware, which we then send for inspection to an online service called VirusTotal. VirusTotal hosts (at the time of writing) 46 signature-based detection engines from different AV vendors. Each AV either detects a malware (in which case it assigns a label to malware it detects) or fails to detect it. We then continued sending the collected malware to the signature-based detection engines in VirusTotal over a period spanning November 2012 to March 2013 to observe whether we see evidence of regressions, and also whether we see changes with which the different vendors identify the same malware over time. We then analyzed the observations to also check for possible signs of correlation, or cause and effect relationships between regressions and label changes. We report the results of this analysis in this paper.

This research has been motivated by the need to gain a better understanding about the reliability of the software engineering practices that different vendors may be employing. As already stated, the whole analysis has been done using externally observable properties of the AV products: namely whether they detect a given malware, and, if they detect it, what label they assign to that malware. For software architects and managers that need to make decisions on what AV products to choose, this analysis provides other selection criteria that they may decide to use to help them make a better decision and reduce risks of making a sub-optimal choice for their chosen deployment environment. We acknowledge that there are limitations to the VirusTotal service (e.g. VirusTotal only provides the signature-based detection engines of the AV products, rather than other capabilities, such as heuristics, and reputation based detection components), but nevertheless each of the vendors is being compared against a single common component only: namely the signature-based detection engine. This makes the comparison across vendors fair. To the best of our knowledge this is the first study that has systematically analyzed AV regressions and label changes from many different vendors.

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

For the sake of brevity, unless otherwise specified, in the rest of the paper we shall use the term AV to refer to the signature-based detection engines of the AV vendors that are provided by the VirusTotal service. The signature-based detection engine is the component that almost all AV products contain and which uses predefined rules to detect, identify and label a malware.

The rest of the paper is organized as follows: Section II contains a summary of related work in AV detection analysis and regression testing; Section III provides a description of the experimental architecture; Section IV provides an analysis of the results of regressions and label changes of the AVs for the malware in our collection period; Section V provides a discussion of the results and finally Section VI contains conclusions and provisions for further work.

II. RELATED WORK

There are many studies that perform analysis of the detection capabilities and rank various AVs. An interesting analysis of "at risk time" for single AVs is given in [4]. In this paper the authors analyzed how long it takes for different AV vendors to detect a malware. As we stated previously, there are numerous sites that report rankings and comparisons of AVs (see footnote 1 for links to these sites).

There have also been studies to assess the benefits in improved malware detection from using more than one diverse AV. An initial implementation of an architecture called Cloud-AV, which utilizes multiple diverse AVs to detect the malware was given in [5]. The authors in [5] also describe an empirical analysis of the benefits of diversity based on the deployment of Cloud-AV at the University of Michigan network. Some of the authors of this paper have also performed large-scale studies on the detection capabilities of diverse AVs which have been published in [1-3]. In this earlier research we utilized the VirusTotal service for the analysis. For the research reported in two of these studies [1, 2] the malware samples were collected by a real world honeypot deployment - SGNET [6, 7], whereas the malware samples reported in [3] were collected using the same infrastructure described in Section III of this paper.

Regression testing is a well known technique in software testing [8]. The aim of this kind of testing is to ensure that changes or updates in the software do not introduce new faults. A good introduction as well as examples of regression testing tools can be found here[2].

Over the past several years, researchers and practitioners have used honeypots to learn about attacks and attackers. These systems can be categorized as security tools *whose value lies in being probed, attacked, or compromised* [9]. These carefully monitored systems allow security researchers to attract hackers, analyze their actions and profile them [10].

Honeypot systems can be found at different scales: from a single host to more complex honeypot networks. These networks, also called honeynets, can be deployed on few IP addresses within a local network. The project Leurre.com [11], SGNET [6, 7] and the honeynet initiative from CAIDA [12] are examples of distributed honeypot networks in different locations.

III. DESCRIPTION OF THE EXPERIMENTAL ARCHITECTURE

Dionaea[3], a low interaction honeypot used to emulate common vulnerable services, has been deployed on a distributed honeypot network architecture. Dionaea captures malicious payload submitted by attackers during the exploitation of exposed network services. Dionaea presents several advantages compared to a high interaction honeypot: 1) it emulates many well-known vulnerabilities and protocols, 2) it is easier to maintain than a high interaction honeypot, and 3) the level of interaction is sufficient to allow successful malicious payload injections.

The 1,136 public IP addresses dedicated to Dionaea were distributed over 7 different networks and 4 geographic locations: France, Germany, Morocco and the USA. Table I shows the repartition of the IP addresses per subnet.

TABLE I. THE GEOGRAPHICAL LOCATIONS OF IP ADDRESSES

Subnet type	Country	Number of IPs
University 1	France	2
University 2	Germany	9
University 3	Morocco	2
Company	United States	3
University 4	United States	1044
University 5	United States	55
University 6	United States	21

This study does not intend to compare the malware collected on the different networks or locations. The different subnets do not have the same size. For instance, because of some of the author's affiliation, University 4 has provided a significantly higher number of IP addresses and allowed to deploy a larger number of honeypots. Note also that the different organizations involved in this distributed architecture apply different security policies. As a consequence each network is not protected in the same way.

Dionaea's default configuration exposes several well-known vulnerabilities of common Internet services (Figure 1) such as *http*, *ftp*, *smtp*, MS SQL, MySQL, as well as Microsoft Windows and VOIP protocols. Because of the nature of the exposed vulnerabilities, Dionaea essentially captures Windows Portable Executable (PE) files[4], the executable file format used on Windows platforms.

Dionaea waits for attackers to inject malicious payloads known as shellcodes by exploiting one of the service's vulnerabilities. The shellcodes are evaluated using *libemu*[5], a C library able to detect and execute shellcodes using the GetPC heuristics [13]. The shellcode profiling allows Dionaea to act upon three possible intentions: 1) providing a remote shell to the attacker by opening a network socket on the targeted system, 2) downloading a file from a remote

[2] http://www.softwaretestinghelp.com/regression-testing-tools-and-methods/

[3] http://dionaea.carnivore.it

[4] http://msdn.microsoft.com/en-us/magazine/cc301805.aspx

[5] http://libemu.carnivore.it

location using ftp, http or SMB protocols, or 3) executing an existing binary file on the local file system of the target host. Dionaea executes multi-staged shellcodes in a virtual machine using libemu to infer their final intent.

Binary files can be captured in different ways: ftp and http downloads, and downloads occurring during the shellcode executions. They can have different formats. The Unix command *file*[6] allows the file format to be identified. Empty and ASCII files were automatically discarded from this analysis as well as the data format that describes unknown binary files. Hence only Microsoft Windows PE files and MS DOS executable files were collected for this study.

```
Starting Nmap 5.21 (http://nmap.org) at 2012-11-12 22:24 EST
Nmap scan report for XXX.XXX.XXX.XXX
Host is up (0.039s latency).
Not shown: 986 closed ports
PORT     STATE    SERVICE
21/tcp   open     ftp
25/tcp   filtered smtp
42/tcp   open     nameserver
80/tcp   open     http
135/tcp  open     msrpc
443/tcp  open     https
445/tcp  open     microsoft-ds
554/tcp  open     rtsp
1433/tcp open     ms-sql-s
2222/tcp filtered unknown
3306/tcp open     mysql
5060/tcp open     sip
5061/tcp open     sip-tls
7070/tcp open     realserver
Nmap done: 1 IP address (1 host up) scanned in 9.18 seconds
```

Figure 1 - Open ports in Dionaea

A test version of Dionaea has been deployed to try its different functionalities. Dionaea names captured binary files after their MD5 hashes and logs the capture or malware submission into an SQLite database. Each entry of the submission database contains:

- The MD5 hash of the binary,
- A capture timestamp,
- The source and destination IP addresses,
- The source and destination ports,
- The protocol exploited,
- The transport protocol (TCP or UDP), and
- The URL used to download the binary file.

Our tests of Dionaea showed that the same binary file could be submitted several times by different originating hosts.

An instance of Dionaea has been deployed on each of the seven subnets that consists of a Linux Virtual Machine running the low interaction honeypot. The captured malware and their submission information are merged and centralized on a single server to facilitate the analysis.

The malware centralization and submission process consists of two steps. The first script written in Perl is executed every day at midnight to download the binary files from the different virtual machines running Dionaea to the main server. This script also fetches the SQLite database that contains the malware submission information and merges them into a single database. Then the whole repository is submitted to VirusTotal. VirusTotal returns a key when a file is successfully submitted. This scan key is built from the binary's SHA1 hash and the submission timestamp. The script keeps track of the scan key to ensure a correct submission of each malware and then later retrieves the VirusTotal analysis.

The second step of the process includes the execution of a second Perl script. By using the scan keys generated by VirusTotal at the submission, the script retrieves the analysis reports from the different AVs. The results sent by VirusTotal are presented in an array. This array contains:

- The response code and response status indicating that the analysis is completed,
- The AV products that have flagged the file as malicious (i.e. detected the malware),
- The total number of AVs used in the analysis, and
- The AVs names, versions and the signatures name.

If the first attempt to retrieve the report for a malware fails, the Perl script will attempt to retrieve it until a response code indicating a completed analysis is received. The script then uploads the content of the report in a MySQL database. Everyday a new database entry is created for each malware. This entry contains the information related to the VirusTotal submission. The AV's names, versions and the malware signature names are also uploaded in various tables and linked together with the file submission.

A limitation of VirusTotal requires a delay of one second between each binary file that is submitted. It takes about one to two seconds to submit a file and its analysis can be executed within a minute depending on the service load. To make sure that all the files are uploaded and analyzed a delay of four hours was set between the submission to VirusTotal and results retrieval.

IV. Analysis of the Results

A. *Descriptive Statistics*

Data collection lasted for 135 days: 10th November 2012 until 24th March 2013. During this period we collected 2,185 malware. The vast majority of the malware (2,174) were Microsoft Windows Portable Executable (PE) files and 11 were MS DOS executables. These malware were sent to VirusTotal where they were examined by up to 46 AV products. We sent the malware on the first day of observation and continued to send them throughout the collection period. However, the total number of datapoints we have is not simply 135 * 2,185 * 46. It is smaller because:

- Not all malware were observed in the first day of collection – we continued to observe new malware throughout the collection period and we cannot send a newly collected malware to older versions of AVs running on VirusTotal;
- VirusTotal may not always return results for all AVs – we are not sure why this is. VirusTotal is a black-box service and its internal configuration is not provided. We presume that each AV is given a certain amount of time to respond and if it doesn't, VirusTotal will not return a result for that AV. There might also be issues

[6] http://darwinsys.com/file/

with a particular AV not being available at the time we submitted the malware for inspection.

A unique "demand" for the purpose of our analysis is a {Malware$_j$, Date$_k$} pair which associates a given malware *j* to a given date *k* in which it was sent to VirusTotal. We treat each of the malware sent on a different date as a unique demand. If all 2,185 malware were sent to VirusTotal on each of the 135 days of data collection, then we would have 2,185 * 135 = 294,975 demands. But as explained before, due to missing data, the number of demands sent to any of the AVs is smaller than 294,975.

If we now associate a given AV *i*'s response to a given malware *j* on a given date *k* then we can consider each of our datapoints in the experiment to be a unique triplet {AV$_i$, Malware$_j$, Date$_k$}. For each of these triplets we have defined a binary score: 0 in case of successful detection, 1 in case of failure. Table II shows the aggregated counts of the 0s and 1s for the whole period of our data collection. We have considered as success the generation of an alert by an AV regardless of the nature of the alert itself.

TABLE II. COUNTS OF DETECTIONS AND FAILURES FOR TRIPLETS { AV$_I$, MALWARE$_J$, DAY$_K$}

Value	Count
0 – detection / no failure	8,812,080
1 – no detection / failure	801,096

Table III contains the failure rates of all the 46 AVs. The ordering is by the failure rate (second column) with the AV with the smallest failure rate appearing first.

The third column in Table III counts the number of "releases" of a given AV recorded by VirusTotal. We presume these are the versions of either the rule set of the detection engine or the release version of the detection engine itself. It seems that different products have different conventions for this. Amongst the three best AVs in our study, Comodo reports 300 whereas Norman and Ikarus report 4 sub-release versions.

The fourth and fifth columns of Table III contain counts of regressions [2]. We count a regression when an AV fails to detect on date *k+1* a malware which it had detected on day *k*. The fourth column contains the number of malware on which a given AV regressed, and the fifth column contains the number of instances of these regressions (since an AV may have regressed more than once on a given malware: alternated between detection and non-detection of a malware several times). We note that even a few AVs, which are in the top ten in terms of the overall detection rates, did have cases of regressions. 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, or the vendor might be attempting to consolidate and streamline the signature-based detection rules (i.e. define a smaller number of more generic rules) to achieve faster detection.

The sixth column contains the counts of signature label changes. We count a signature label change when an AV changes the label with which it identified a malware on date *k+1* compared with how it identified the malware on date *k*.

TABLE III. FAILURE RATES, RELEASE COUNTS, REGRESSION COUNTS, AND LABEL CHANGES COUNTS FOR EACH AV

AV Name	Failure rate	Number of "releases" of the AV in VirusTotal	Count of Malware on which AV regressed	Count of Regression Instances	Count of Label changes
Comodo	0.00017	300	0	0	20
Norman	0.000453	4	39	39	1780
Ikarus	0.000674	4	7	7	376
McAfee-GW-Edition	0.000714	2	117	145	2994
AVG	0.000894	1	0	0	35
Panda	0.001048	1	33	85	114
Symantec	0.001568	5	99	108	58
TheHacker	0.002374	2	6	6	34
AntiVir	0.003159	304	0	0	39
TrendMicro-HouseCall	0.003634	2	537	626	68
VIPRE	0.004398	336	14	14	205
Jiangmin	0.004429	3	4	4	25
Kingsoft	0.005909	5	31	54	429
PCTools	0.006764	2	78	102	547
Agnitum	0.008059	1	140	145	1
BitDefender	0.012567	1	2	2	175
F-Secure	0.01259	3	5	5	2898
GData	0.013964	1	279	298	169
Avast	0.014726	1	3	3	120
F-Prot	0.015667	2	68	68	3
nProtect	0.016076	224	0	0	59
McAfee	0.016744	1	16	27	1784
AhnLab-V3	0.016834	235	75	82	20
Kaspersky	0.017786	1	11	11	10
Sophos	0.017826	10	52	56	58
Emsisoft	0.017951	4	15	15	1014
TotalDefense	0.018539	211	0	0	166
ViRobot	0.01854	2	4	4	20
K7AntiVirus	0.019779	167	12	12	557
TrendMicro	0.020509	7	589	703	53
Microsoft	0.021461	11	1	1	59
ESET-NOD32	0.021693	262	27	27	14
DrWeb	0.023253	6	5	5	38
VBA32	0.024699	9	7	7	13
NANO-Antivirus	0.02703	9	0	0	302
Commtouch	0.028271	2	8	9	855
Rising	0.044548	110	386	394	14
CAT-QuickHeal	0.047723	2	1643	1663	97
SUPERAntiSpyware	0.114503	4	7	7	1
MicroWorld-eScan	0.164515	1	2023	21494	123
eSafe	0.24881	1	18	19	1366
ClamAV	0.356962	1	1	1	4
Fortinet	0.418235	4	1050	1051	94
Antiy-AVL	0.442624	3	48	49	73
Malwarebytes	0.689115	2	465	485	13
ByteHero	0.930788	1	83	1087	0

Even though some of the AVs have very good detection rates (i.e. low failure/non-detection rates) none of them have detected all the malware in our study. We can also see that some AVs have really low detection rates, with the bottom 8 AVs failing to detect more than 10% of all the demands sent to them. We are not certain why this is the case. It could be because the AV vendors are failing to keep their AVs in VirusTotal up to date with their latest signatures, even if their products in commercial installations are up to date (though some of these vendors do seem to be updating their products with new release numbers, as evidenced from column three values), it could be that these products have more advanced anomaly and reputation based features (not available for testing in VirusTotal) or it could be because these AVs genuinely have low detection rates for this dataset.

B. *Regressions Analysis*

Figures 2 and 3 show 3D plots of the regressions for the AV and Date dimensions (Figure 2) and AV and Malware dimensions (Figure 3). Each value in the plot gives the ratio of regressions observed for either a given AV on a given date (Figure 2), or for a given AV on a given malware (Figure 3). The ratio values are shown by the intensity of the color in the plot (with green being low ratio values (close to 0) and red being high (close to 1)), with white areas meaning no regressions have been observed, and black areas representing missing data. We have kept the ordering of the AVs by failure rate as in Table III, with the AVs with lowest failure rates appearing on the left of the plots.

These figures further illustrate the trends for the regressions counts we showed in Table III. We can see that even some AVs with high detection rates regress regularly during our data collection period. So for some AVs these regressions are not just isolated incidents in particular points in time, but rather seem to be regular occurrences. Hence some of these AVs in particular would especially benefit from regression testing.

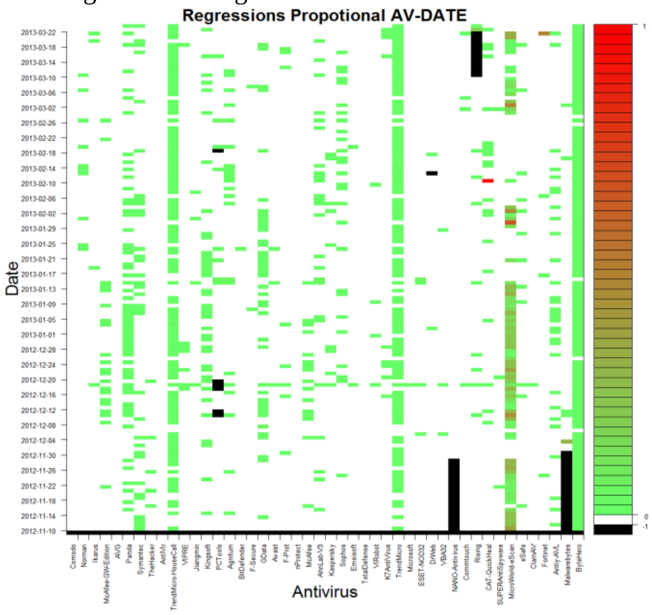

Figure 2 – AV (x-axis), dates (y-axis) and the proportion of malware that have caused a particular AV to regress on a particular date.

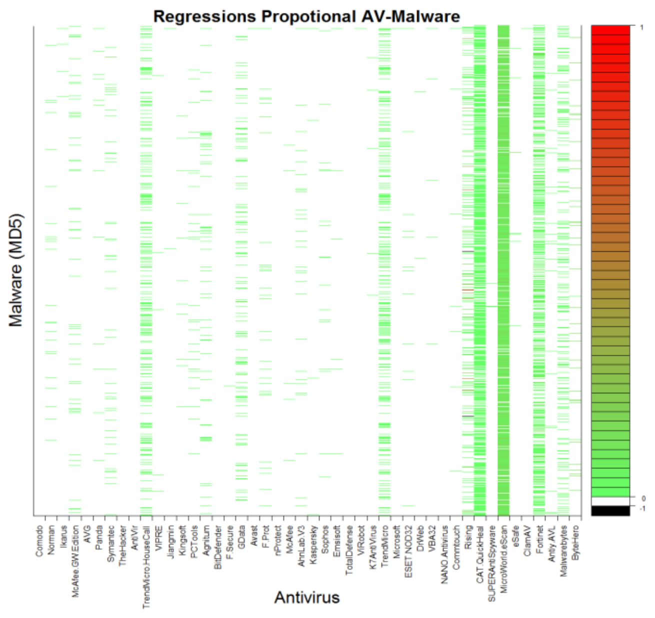

Figure 3 - AV (x-axis), malware (y-axis) and the proportion of days in which a given AV has regressed on a given malware.

C. *Label Changes Analysis*

Table IV is an analysis report from VirusTotal of the malicious sample identified by the MD5 hash fb7ba7e14bafdc97724ffb66d39c2246. The report provides for each AV the label used to identify the malware. In this present case, the malicious program is Conficker[7].

Malicious programs can be classified in different categories depending on their nature such as worms, Trojans, backdoors etc. AVs signature labels can be used to determine these categories. For example, ClamAV uses the following signature label naming convention[8]: `malware_type.family-variant`. When we subjected all these malware to our locally installed copy of ClamAV it classified the malware as follows: 75.5% worms, 20.5% Trojans, 0.4% backdoors or Windows viruses, and 3.6% were unknown to this ClamAV version. Table IV also shows that these categories may vary from one AV to another.

Depending on the AV, the signature labels may follow a specific naming convention. Most of them will include the type of malware, sometimes the platform or operating system impacted, a family name and variant numbers or letters. However, some AVs use generic labels to qualify the malicious sample or just a number.

The analysis of the label changes shows a common behavior: a generic label is first given to an unknown sample when an AV has detected a suspicious behavior using heuristic algorithms. The label is then changed when the threat is clearly identified as a variant of an existing malware or a new malware. For example the AV nProtect label changes occurred when a malicious program was first

[7] http://www.confickerworkinggroup.org/

[8] http://www.clamav.net/doc/latest/signatures.pdf

identified as Win32.Worm.Downadup.Gen and then as Worm/W32.Kido.139680.D.

TABLE IV. VIRUSTOTAL ANALYSIS REPORT FOR CONFICKER MALWARE

AV	*Engine Version*	*Signature Label*
Avast	6.0.1289.0	Win32:Confi [Wrm]
Antiy-AVL	2.0.3.7	Trojan/Win32.Agent.gen
Ikarus	T3.1.1.122.0	Trojan-Downloader.Win32.Kido
Panda	10.0.3.5	W32/Conficker.C.worm
VBA32	3.12.18.2	Worm.Win32.kido.106
TrendMicro-HouseCall	9.500.0.1008	WORM_DOWNAD.AD
Emsisoft	5.1.0.11	Trojan-Downloader.Win32.Kido!IK
CAT-QuickHeal	12	Win32.Net-Worm.Kido.ih.4
SUPERAntiSpyware	4.40.0.1006	Trojan.Agent/Gen
McAfee-GW-Edition	2012.1	W32/Conficker.worm.gen.a
TrendMicro	9.561.0.1027	WORM_DOWNAD.AD
Kaspersky	9.0.0.837	Net-Worm.Win32.Kido.ih
ViRobot	2011.4.7.4223	Worm.Win32.Conficker.165405
Microsoft	1.8601	Worm:Win32/Conficker.C
Jiangmin	13.0.900	Worm/Kido.ml
McAfee	5.400.0.1158	W32/Conficker.worm.gen.a
ClamAV	0.97.3.0	Worm.Kido-96
F-Secure	9.0.16440.0	Worm:W32/Downadup.gen!A
eSafe	7.0.17.0	Win32.Banker
F-Prot	4.6.5.141	W32/Conficker!Generic
AVG	10.0.0.1190	Worm/Downadup
Norman	6.08.06	W32/Conficker.GW
Symantec	20121.1.0.298	W32.Downadup.B
GData	22	Win32.Worm.Downadup.Gen
Commtouch	5.3.2.6	W32/Conficker!Generic
TheHacker	None	W32/Kido.ih
BitDefender	7.2	Win32.Worm.Downadup.Gen
PCTools	8.0.0.5	54815
Sophos	4.80.0	Mal/Conficker-A
DrWeb	7.0.3.07130	Win32.HLLW.Shadow.based
K7AntiVirus	9.147.7516	NetWorm
AntiVir	7.11.40.82	Worm/Conficker.AA
AhnLab-V3	2012.08.20.00	Win32/Kido.worm.165405
Rising	24.24.00.01	Trojan.Win32.Generic.122F15C8
nProtect	2012-08-20.01	Worm/W32.Kido.165405
VirusBuster	15.0.155.0	Worm.Kido!2hIqC2kC9R4
Comodo	13286	NetWorm.Win32.Kido.A
TotalDefense	37.0.10037	Win32/Conficker
VIPRE	12706	Worm.Win32.Downad.Gen (v)
ESET-NOD32	7399	a variant of Win32/Conficker.X

The opposite behavior is sometimes observed as well: a specific name is given to a malicious sample and then changed to a more generic category. For example the AV Symantec changed a signature label from Trojan.ADH.2 to a more generic one Packed.Generic.382.

Few changes introduce new variants numbers or new threat types.

Table V contains counts of malware per AV grouped by the number of label changes they had in our dataset. We grouped them in categories representing 0, 1, 2, 3, 4 or 5 or more label changes. We kept the ordering of the AVs by failure rate as in Table III. Note that in Table III we had shown the overall number of label changes per AV, whereas Table V shows the number of malware grouped by the count of label changes a particular AV exhibited (e.g. Comodo had 20 label changes in total: 18 malware with 1 label change, and 1 malware with 2 label changes).

TABLE V. COUNTS OF MALWARE PER AV GROUPED BY THE NUMBER OF LABEL CHANGES THEY HAD IN OUR DATASET (0 TO 5 OR MORE)

AV Name	*0*	*1*	*2*	*3*	*4*	*5+*
Comodo	1862	18	1	0	0	0
Norman	382	1218	281	0	0	0
Ikarus	1535	321	21	3	1	0
McAfee-GW-Edition	1048	204	199	12	190	228
AVG	1857	19	0	4	1	0
Panda	1848	20	3	3	2	5
Symantec	1840	25	15	1	0	0
TheHacker	1849	30	2	0	0	0
AntiVir	1460	258	156	3	3	1
TrendMicro-HouseCall	1833	32	13	2	1	0
VIPRE	1750	70	53	3	5	0
Jiangmin	1860	17	4	0	0	0
Kingsoft	1469	397	13	2	0	0
PCTools	1366	490	20	4	0	1
Agnitum	1880	1	0	0	0	0
BitDefender	1793	51	24	7	0	6
F-Secure	440	17	1421	1	0	2
GData	1793	51	24	7	0	6
Avast	1765	112	4	0	0	0
F-Prot	1878	3	0	0	0	0
nProtect	1828	49	2	2	0	0
McAfee	1014	174	152	8	100	433
AhnLab-V3	1866	10	5	0	0	0
Kaspersky	1872	8	1	0	0	0
Sophos	1846	12	23	0	0	0
Emsisoft	872	1005	3	1	0	0
TotalDefense	1717	162	2	0	0	0
ViRobot	1863	17	0	1	0	0
K7AntiVirus	1485	242	150	1	3	0
TrendMicro	1829	17	18	2	14	1
Microsoft	1822	59	0	0	0	0
ESET-NOD32	1868	12	1	0	0	0
DrWeb	1849	29	0	3	0	0
VBA32	1868	13	0	0	0	0
NANO-Antivirus	1717	80	30	54	0	0
Commtouch	1346	253	254	18	10	0
Rising	1869	11	0	1	0	0
CAT-QuickHeal	1821	26	33	1	0	0
SUPERAntiSpyware	1880	1	0	0	0	0
MicroWorld-eScan	1799	66	9	5	0	2
eSafe	1769	7	6	0	1	98
ClamAV	1877	4	0	0	0	0
Fortinet	1810	48	23	0	0	0
Antiy-AVL	1808	73	0	0	0	0
Malwarebytes	1870	9	2	0	0	0

Figures 4 and 5 show plots of the label changes for the AV and Date (Figure 4) and AV and Malware (Figure 5).

We can see that AVs seem to change the labels with which they identify the malware fairly regularly. Even the AVs with high detection rates (that appear towards the left hand side of the plots) seem to frequently change the labels with which they name the malware. So there is a continuous activity in the naming of these malware that seems to be happening for many of these AVs.

Note that Figure 5 shows a lot of missing data for the right-most 6 AVs. As we saw in Table III these are AVs that have very high failure rates. These AVs fail to detect large proportions of malware. Since we can only count label changes when AVs detect malware, we have no data for label changes for large proportions of these malware – hence the large black areas in the plot.

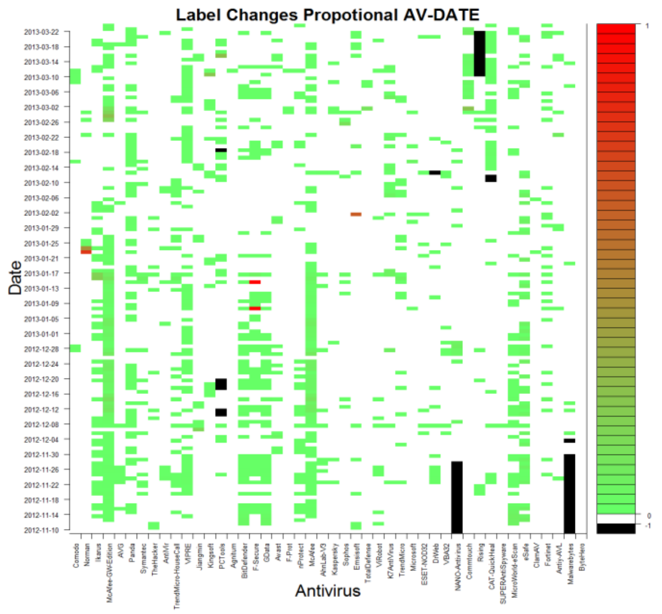

Figure 4 - AV (x-axis), dates (y-axis) and the proportion of malware for which a particular AV changed labels on a given date.

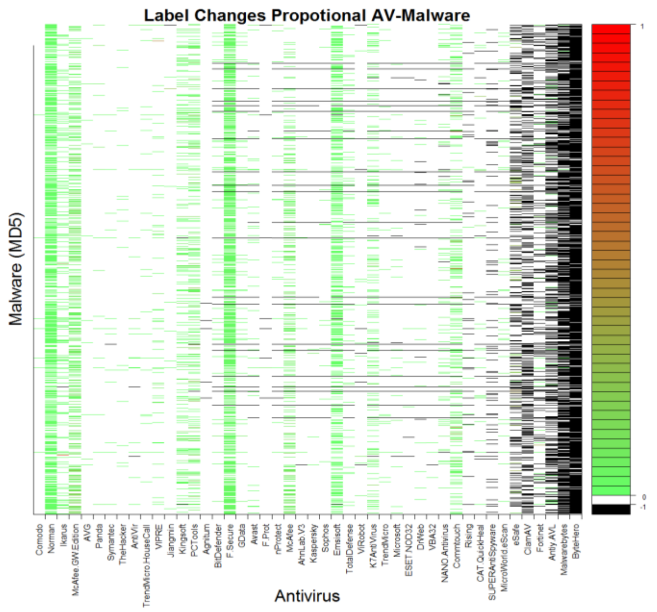

Figure 5 - AV (x-axis), malware (y-axis) and the proportion of days a given AV has changed labels for a given malware detection.

D. Correlation of Regressions and Label Changes

So far we have looked at the trends of label changes and regressions separately. In this section we will analyze the relationships between these two aspects.

Figures 6 and 7 show the plots for the combined ratios of label changes and regressions in the AV-date and AV-malware dimensions respectively. The coloring rules are as follows:

- Black - No data: in the combined plots no data means we have missing data for either regressions or label changes (or both);
- White - No Regressions and no Label changes;
- Yellow - Regressions but no Label changes;
- Green – No Regressions but Label changes;
- Red – Regressions and Label changes.

We see several areas in the graphs colored red, meaning proportions of both regressions and label changes on those days for the same AV (in Figure 6) or both regressions and label changes for the same instances of malware for a given AV (in Figure 7). These are especially prominent for the TrendMicro, McAfee and the MicroWorld AVs (from Figure 6). In the AV-malware plot the red patches are less pronounced partially because there are more malware being plotted in the y-axis (2185) compared with the dates (135).

To study the correlations between label changes and regressions in more detail we calculated the Pearson product-moment correlation coefficients for each AV. The Pearson product-moment correlation coefficient for two arrays X (label changes) and Y (regressions) is as follows:

$$Correl\ (X, Y) = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

The correlation coefficient values are shown in Figure 8. We cannot calculate correlation coefficients for seven AVs which exhibited either no regressions, or no label changes, or neither (Comodo, AVG, AntiVir, nProtect, TotalDefense, NANO-Antivirus, ByteHero) so they are not shown in Figure 8 (or Figure 9). From Figure 8 we see that three AVs have correlation coefficients of 0.7 or higher which suggests high positive correlation between regressions and label changes for them. Note that Commtouch, which appears as the AV with the highest correlation coefficient, did not seem to exhibit significant regressions or label changes results, as shown in Figures 6 and 7. However, on closer inspection it was seen that for this AV there were long periods when there were no label changes or regressions, and then short bursts of label changes, which were coincident with a few cases of regressions. This results in a strong correlation coefficient seen in Figure 8.

We also calculated correlation coefficients for each AV when we removed the "white" areas from the calculations: i.e. instances where we had neither label changes nor regressions. This is shown in Figure 9. The AVs which exhibit positive correlations tend to be the same as the ones in Figure 8 (though the ordering based on CC values changes).

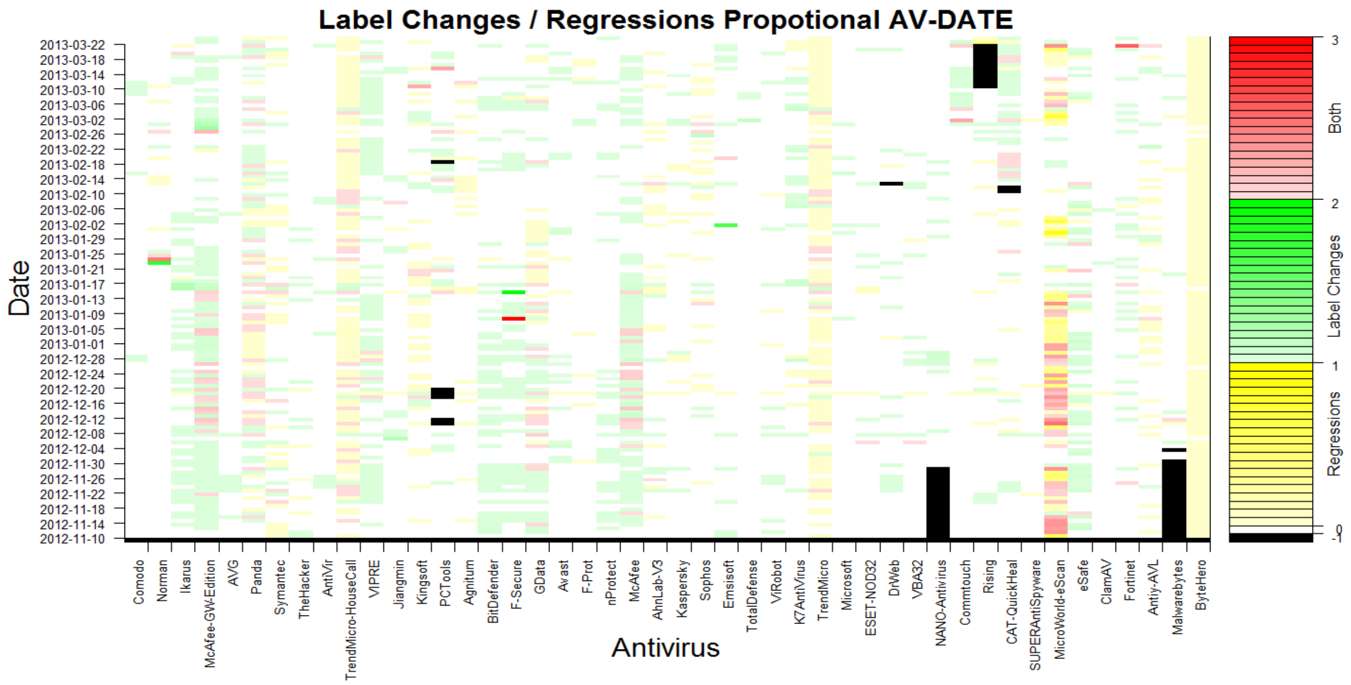

Figure 6 - AV (x-axis), dates (y-axis) and proportion of regressions only (yellow), label changes only (green), or both label changes and regression (red). White color - no regressions or label changes; black – no data for either label changes or regressions, or both.

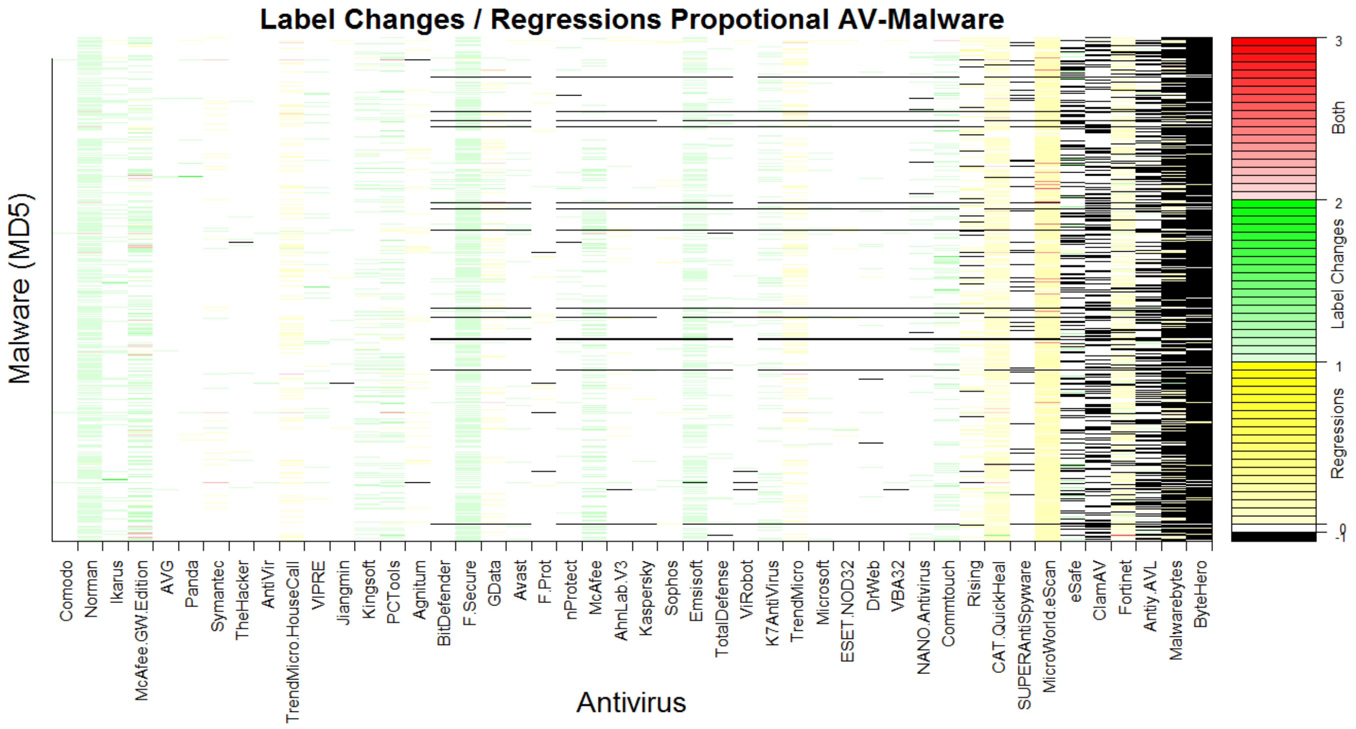

Figure 7 - AV (x-axis), malware (y-axis) and the proportion of regressions only (yellow), label changes only (green), or both label changes and regression (red). White color - no regressions or label changes; black – no data for either label changes or regressions, or both.

We do see a reversal from a mildly positive to a mildly negative correlation for some of the AVs - this is because for these AVs there were few cases of regressions and label changes, and hence removing the white areas (i.e. two 0 values, indicating positive correlation), leaves only a few values on which to calculate the CCs, which would generally shift the CC value towards a negative correlation side. And we also see a more pronounced negative correlation shown for some of the other AVs. This means that for these AVs, on the same dates, when we see label changes we tend not to see regressions and vice versa (again removal of the white areas makes this negative relationship stronger).

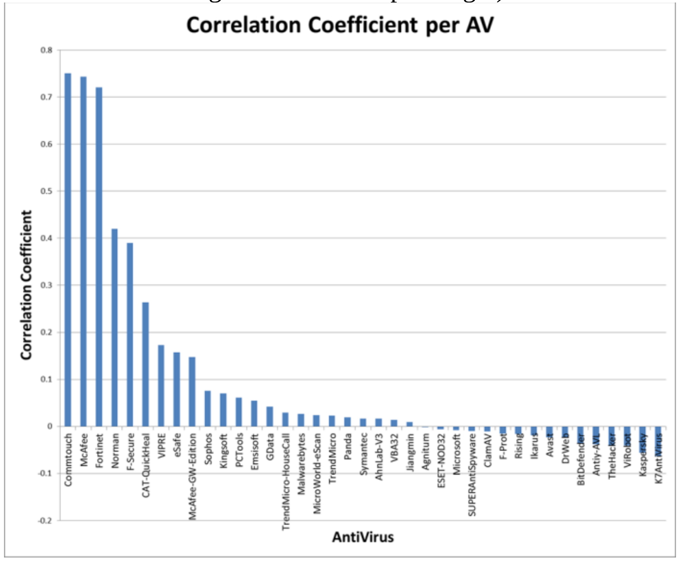

Figure 8 – Correlation coefficients per AV. The grouping of label changes and regressions for each AV was done per date.

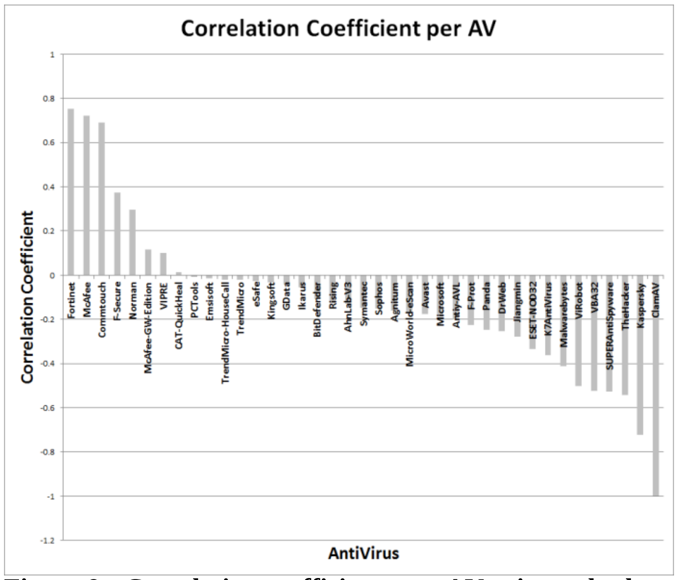

Figure 9 - Correlation coefficients per AV using only the data where there is either a regression or label change. The grouping of label changes and regressions for each AV was done per date.

V. DISCUSSION

The preliminary results look intriguing. For some of these AVs there does seem to be a relationship between label changes and regressions. This can be visually inspected from the joint plots of regressions and label changes, but also confirmed from the correlation coefficients. The reason for these coincident values might be because of lack of regression testing prior to the updating of the malware detection rules by these vendors. Hence some of these vendors may benefit from more systematic use of regression testing before releasing updates to their malware detection rulesets.

On the other end of the spectrum we also see that there are some AVs (Comodo, AVG, AntiVir, nProtect, TotalDefense, NANO-Antivirus) for which we have observed label changes but no regressions at all. These AV vendors may have good regression testing practices implemented in house or very good software engineering practices implemented that lead to fault avoidance during the updating of the rulesets.

When we calculated correlation coefficients using just the data where we had either regressions or label changes (cf. Figure 9) we see that there are also some AVs that exhibit strong negative correlations when we group the label changes and regressions per date. This might be due to a delayed effect: we see evidence that, in some cases, on the same date we either see label changes or regressions, but not both. But it might mean that progressive label changes do have a delayed effect on regressions.

VI. CONCLUSIONS

In this paper we presented an analysis of the relationship between two externally observable properties of AV products: *regressions* of the AVs in their detection capability (i.e. their (reoccurring) failure to detect a malware which they had successfully detected in the past); and the *change of labels* with which the AVs mark detected malware.

We ran an experimental campaign over a 135 day period with 2,185 malware samples collected in our distributed low-interaction honeypots environment, which we sent to 46 signature-based detection engines of AVs provided by the VirusTotal service.

To the best of our knowledge this is the first paper that has analyzed the relationship between regressions and malware label changes for AVs.

Our main findings can be summarized as follows:

- For some of these AVs there seems to be a relationship between label changes and regressions: we saw either a high Pearson product-moment correlation coefficient value, or a large number of dates in which both regressions and label changes were observed. We speculate that the reason for these coincident values might be because of a lack of regression testing prior to the updating of the malware detection rules by these vendors. We therefore postulate that some of these vendors may benefit from more systematic use of regression testing before releasing updates to their malware detection rulesets.

- There are some AVs for which we have observed label changes but no regressions. These AV vendors may have good regression testing practices implemented in house, do not frequently change the labels for the malware, or they have very good software engineering practices implemented that lead to fault avoidance during the updating of the rulesets (or a combination of the above).

It is difficult to conclude with high confidence that the *causes* of regressions or lack of them (i.e. the *effect*) are the software engineering practices and/or lack of regression testing that the AV vendors employ. This is because we are only dealing with the externally visible behavior of these products (i.e. whether they detect a malware or not in time, and what labels they attach to those malware they detect). More confidence about the cause and effect relationships could be gained if we had insight into the vendors' practices. But we hope our analysis and results will provide sufficient motive for some of these vendors to enhance their in-house software engineering and regression testing procedures to improve their overall malware detection capabilities.

The main limitation of our study which prevents us from making more general conclusions is that VirusTotal only provides the signature-based detection engines of these vendors. We don't know whether in cases where there is a regression of the signature-based detection engine if some other part of the AV product detects that malware. Hence further work is required using the full capabilities of these products to perform a more complete analysis of the detection capabilities.

Other provisions for further work include:

- Studying the interplay between the false positive, false negative rates on the one hand and the rates of regressions and label changes on the other. All our analysis so far has been with malware samples, which means we cannot get any measurements on false positive rates. It would be interesting to observe what effect the false positive rate has on the dynamics of regressions and label changes.
- Studying the patterns of label changes: in Table V we saw that some AVs change the labels for a malware multiple times to potentially multiple different names. It will be interesting to observe if there are specific patterns that these changes have over time (e.g. changing from a more generic name, to something more specific etc.), and what impact this has, if any, on the overall detection rates and rate of regressions.

ACKNOWLEDGEMENT

This work was supported in part by the City University London Pump Priming fund via grant "Empirical Analysis of Diversity for Security", and the SeSaMo project, supported by the Artemis JU, and the United Kingdom Technology Strategy Board (ID 600051 and 600052). The authors thank Gerry Sneeringer and the Division of Information Technology at the University of Maryland for allowing and supporting their research.

REFERENCES

[1] 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.

[2] 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.

[3] Cukier, M., I. Gashi, B. Sobesto and V. Stankovic. "Does Malware Detection Improve with Diverse Antivirus Products? An Empirical Study". in 32nd International Conference on Computer Safety, Reliability and Security (SAFECOMP). Toulouse, France., 2013.

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

[5] 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.

[6] Leita, C. and M. Dacier. "Sgnet: A Worldwide Deployable Framework to Support the Analysis of Malware Threat Models". in 7th European Dependable Computing Conference (EDCC). Kaunas, Lithuania, p. 99 - 109, 2008.

[7] 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.

[8] Myers, G., C. Sandler, T. Badgett and T.M. Thomas, The Art of Software Testing. 2nd ed, Wiley. 2004

[9] Spitzner, L., Honeypots: Tracking Hackers, Addison-Wesley Longman Publishing Co., Inc. 2002

[10] Ramsbrock, D., R. Berthier and M. Cukier, "Profiling Attacker Behavior Following Ssh Compromises", in Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. 2007, IEEE Computer Society. p. 119-124.

[11] Pouget, F., M. Dacier and V.H. Pham, "Leurre.Com: On the Advantages of Deploying a Large Scale Distributed Honeypot Platform", in ECCE'05, E-Crime and Computer Conference. 2005: Monaco.

[12] Vrable, M., J. Ma, J. Chen, D. Moore, E. Vandekieft, A.C. Snoeren, et al., "Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm". SIGOPS Oper. Syst. Rev., 39(5): p. 148-162. 2005.

[13] Polychronakis, M., K.G. Anagnostakis and E.P. Markatos, "Comprehensive Shellcode Detection Using Runtime Heuristics", in Proceedings of the 26th Annual Computer Security Applications Conference. 2010, ACM: Austin, Texas. p. 287-296.