Evaluating Detection and Treatment Effectiveness of Commercial Anti-malware Programs

Jose Andre Morales, Ravi Sandhu and Shouhuai Xu
Institute for Cyber Security
University of Texas at San Antonio

Abstract

Commercial anti-malware programs consist of two main components: detection and treatment. Detection accuracy is often used to rank effectiveness of commercial anti-malware programs with less emphasis on the equally important treatment component. Effectiveness measures of commercial anti-malware programs should consider equally detection and treatment. This can be achieved by standardized measurements of both components. This paper presents a novel approach to evaluate the effectiveness of a commercial anti-malware program's detection and treatment components against malicious objects by partitioning true positives to incorporate detection and treatment. This new measurement is used to evaluate the effectiveness of four commercial anti-malware programs in three tests. The results show that several anti-malware programs produced numerous incorrectly treated or untreated true positives and false negatives leaving many infected objects unresolved and thereby active threats in the system. These results further demonstrate that our approach evaluates the detection and treatment components of commercial anti-malware programs in a more effective and realistic manner than currently accepted measurements which primarily focus on detection accuracy.

1 Introduction

Commercial anti-malware programs (CAmps) are the premier line of defense against malicious threats for desktops, laptops, and handheld devices. Consumers expect CAmps to provide complete security protection, accurate detection and effective treatment of malicious objects. Since most consumers are not computer security experts, automatic treatment of malicious objects is preferred to requesting the consumer to decide appropriate treatment. Consumers place their trust in a CAmp to keep their system safe and assume a malicious object is detected and treated when informed of its presence. Comprehensive testing is necessary to evaluate the effectiveness of a CAmp in detecting and treating malicious objects.

A CAmp consists of two main components: the detection component and the treatment component with the assumption that a malicious object is always detected first and then treated. Evaluations of CAmp effectiveness have historically focused primarily on the detection component through black box testing with the output being standardized as false positives, false negatives, true positives and true negatives. Of these measurements, false positives, false negatives and true positives are often considered most important and are integral to ranking CAmp effectiveness. Unfortunately, the treatment component is often not evaluated with the same rigor as the detection component. A popular approach in evaluating the treatment component is a comparison of which malicious objects were detected by a specific CAmp against a list of objects known to be infected by a specific piece of malware [1]. In this approach the treatment is assumed to have been performed successfully and is not verified. Also not considered are detected malicious objects with ambiguous log file labels making it unclear if the malicious object was treated or not. Finally, a CAmp requesting a consumer to choose treatment for a malicious object, a popular approach of many CAmps, is also often not considered. This is dangerous since a consumer can choose the wrong treatment or not choose a treatment at all either out of fear or insufficient knowledge of the correct action to take. When these situations arise during treatment, a system can be left with active malware still executing while the consumer seemingly believes that the system is safe. To avoid these problems, both the detection and treatment components of a CAmp should be evaluated with the same rigor and results reported in a standardized form.

In this paper, we present a new approach to evaluating the effectiveness of a commercial anti-malware program's detection and treatment components against malicious objects. We propose a new standard of reporting evaluation results that incorporates the output of both components. We use the current standard of false positives, false negatives, true positives and true negatives [18]. We further divide true positives into three finer-grained partitions which incorporate the results from the treatment component of a CAmp. We tested four CAmps with three specific tests representing realistic scenarios a user may face when dealing with malware in a system. Each test reports output using our standardized form. Our analysis was primarily based on the log files produced by each CAmp. Our results revealed several instances of four cases: true positives receiving automatic treatment, true positives not receiving treatment, true positives whose treatment is chosen by the consumer with the option to choose incorrectly or not choose at all, and false negatives. The last three cases allowed many undetected and untreated malicious objects to remain as active threats in a seemingly safe to use system.

2 Related Work

A broad corpus of research exists in the current literature presenting different approaches to detecting malware. This is a small sampling of that corpus: [5, 6, 7, 8, 10, 11, 13, 14, 15, 17, 20, 21]. This corpus of research primarily focuses on detection accuracy and not on the important issue of treatment once a detection has occurred. Approaches to perform and standardize testing of anti-malware programs [1, 2, 3, 9, 12, 16] address the general concepts to make testing reliable, transparent and most importantly produce standardized output. Unfortunately, they do not present a method to verify treatment and do not present a formal methodology to represent treatment results in a standardized form. In this paper, we evaluate both detection and treatment effectiveness of commercial anti-malware programs against malicious objects. We present a formal methodology which allows the test output to be expressed in a standard form and we verify treatment that occurred during evaluation testing.

3 Evaluation Methodology

To evaluate the effectiveness of a commercial anti-malware program (CAmp) against malicious objects, we define a function $A()$ with input S where $A()$ is a CAmp and S is any object of a computer system that is accepted as input by $A()$ for the detection and treatment of malicious objects. S can be a single object such as a file saved on disk or a currently executing process or consist of multiple objects such as a group of files or a complete operating system in an infected or clean state. $A(S)$ has four outputs: false positive (FP), false negative (FN), true positive (TP) and true negative (TN). TP and FN measure effectiveness of $A()$ against malicious objects while TN and FP measure correctness of $A()$ with benign objects. $A(S)$ consists of two subcomponents: the detection sub-component $A_D()$ and the treatment sub-component $A_T()$. We assume $A_D()$ always occurs before $A_T()$ and partial output of $A_D()$ serves as input for $A_T()$.

In this paper, we will focus exclusively on two outputs of $A()$: TP and FN thereby measuring effectiveness of $A()$ with respect to malicious objects. Our experiments will be designed to ensure $FP = 0$ and $TN = 0$. We do by ensuring that S consists entirely of malicious objects. Thereby we can set $S = TP + FN$ which establishes all objects of S as malicious and excludes all benign objects.[1] We design three evaluation tests which guarantee $FP = 0$ and $TN = 0$ to the extent possible within intrinsic experimental error. An FN occurs when a malicious object is identified as benign and a TP occurs when a malicious object is correctly identified as malicious. The input of $A_D()$ is S and the output of $A_D()$ is (FN, FP, TN, TP) of which TP, FP serves as the input of $A_T()$. With $FP = 0$ the only input to $A_T()$ is TP.

To measure the effectiveness of $A_T()$ with input TP, we will define the output by partitioning TP into three subgroups representing the three cases where treatment was performed automatically, by user option or not performed at all. First we assume $A_T()$ is either capable or incapable of providing treatment to each malicious object in TP. There are three basic types of treatment: disinfection, where the malicious content is removed from an object; deletion, where a malicious object is permanently erased from a system; and quarantine, where a malicious object is isolated from the system and disallowed interaction with the system and its users. If $A_T()$ can provide treatment for a malicious object, the treatment will occur either automatically or by requesting the user to choose an appropriate treatment option. Therefore, the following three mutually exclusive subgroups of TP are defined: automatic

[1] With some overload of notation for convenience and readability we understand $S = TP + FN$ to denote both that (i) S is the union of TP and FN when these are considered as sets, and (ii) $|S| = |TP| + |FN|$ when we are considering numbers. This notational convenience works since TP and FN are mutually exclusive and carries over to other similar equations with disjoint sets.

treatment (TP_A), user option treatment (TP_O) and no treatment (TP_N). In context of $FP = 0$ these three subgroups comprise the output of $A_T(TP)$ and $TP = TP_A + TP_O + TP_N$.

We construct our definition of the function $A()$ with input S to evaluate the detection and treatment effectiveness of a commercial anti-malware program (CAmp) against malicious objects as follows:

$$A(S) \equiv \left\{ \begin{array}{l} A_D(S) = (FN, FP, TN, TP) \\ A_T(TP, FP) = (TP_A, TP_O, TP_N, FP) \end{array} \right\} \tag{1}$$

and

$$A(S) = (FN, FP, TN, TP_A, TP_O, TP_N) \tag{2}$$

Equation (1) is the relation between $A(S)$ and its subcomponents $A_D()$ and $A_T()$. We assume an object receives treatment from $A_T()$ if and only if the object is first determined as malicious by $A_D()$ and therefore is part of TP or FP. In equation (2) the output TP is replaced by (TP_A, TP_O, TP_N) which allows $A(S)$ to output in a standardized form the detection and treatment effectiveness of a CAmp against malicious objects.

Using equation (2), we consider a CAmp $A()$ with input S to be effective in detection and treatment of malicious objects when the following properties are true for the output of $A()$: minimal or no malicious objects are classified as FN, TP_O, or TP_N and all or a majority of malicious objects are classified as TP_A. We formalize these properties as follows in experiments where $FP = 0$ and $TN = 0$ so that $S = TP + FN$:

$$\frac{TP}{S} \quad and \quad \frac{FN}{S} \tag{3}$$

$$\left(\frac{TP_A}{S} \; \frac{TP_O}{S} \; \frac{TP_N}{S} \right) \quad and \quad \frac{FN}{S} \tag{4}$$

Equation (3) is the traditional measure of TP and FN which evaluates effectiveness of $A_D()$. Equation (4) evaluates both A_D and A_T by containing the newly defined subgroups for TP and is a refinement of Equation (3). To evaluate separately, Equation (3) evaluates detection and (TP_A, TP_O, TP_N) ratios from Equation (4) evaluates treatment. Effective detection of malicious objects is achieved with high TP and low FN. Effective detection and treatment of malicious objects is achieved with high TP_A and low FN. The most desirable output is for all or most identified malicious objects to be grouped as TP_A which represents detected malicious objects which received automatic treatment. TP_O should be minimized since asking a typical user to choose treatment for a malicious object can result in making the wrong decision or choosing not to make any decision. TP_N output should be completely avoided or minimized since these malicious objects did not receive treatment. FN is the most undesirable output since these are undetected malicious objects. These scenarios can leave several undetected and untreated malicious objects active in the system. In summary, a CAmp which maximizes TP_A and minimizes FN, TP_O, and TP_N will achieve the highest level of effectiveness in the detection and treatment of malicious objects.

4 Evaluation Tests

Using equation (2), three experiments (which we also call tests) were designed to evaluate how effective the detection and treatment components of a CAmp are in realistic scenarios which a consumer may face on a system when dealing with malware. Test 1 was the simplest and required each CAmp to scan a folder which contained malware samples. In test 2 each CAmp was installed with the system in a clean state following which execution of known malware was attempted with the CAmp enabled. In test 3 known malware was executed for 3 minutes with the system in a clean state but with no installed CAmp. Subsequently an individual CAmp was installed and required to do a full system scan. Below we will describe each test in detail, its purpose, setup, environment, implementation and results followed by a discussion.

Test Setup and Execution. All three tests were designed so that the input S consists entirely of malicious objects. Thereby $FP = 0$, $TN = 0$ and $S = TP + FN$. All three tests were performed on VMWare Workstation running Windows XP-SP2. A set of four CAmps were used for testing and their log files were analyzed to calculate the results. Each CAmp was a free trial version which was installed and fully updated before testing. All instructions given by the CAmp during installation, detection and treatment were followed. The malware used in the tests were drawn from 974 samples downloaded from the 27 October 2009 upload on the CWSandBox malware repository [19]. The specific date was arbitrarily chosen to be recent, but not too recent. Recency was required so that the malware had not yet become extinct in the wild. Choosing a not too recent date gave the CAmp maintainers time to incorporate detection and treatment of known malware. This set consisted of network worms, peer-to-peer worms, email viruses, rootkits, bots, password stealers, malware downloaders, backdoors, adware and spam generators. All 974 samples

1st malware set	
Trojan.Pasta.anq - 2574eda157245099b0ab2dbc1be2d980	Backdoor.Poison.xtr - 37e8695cc7be98a6ae637c240f09d6c0
Trojan.Buzus.lba - 58242cb6fbf79d7e7ea616e17acf7e11	Virus.Xorala - 7018d9ed260232cd4983ed4f4b59a9c6
2nd malware set	
Net-Worm.Allaple.e - 75da1173c9325b926a58d83ac4949915	Packed.Krap.n - 75e3fc3b0cf524293c8bef77e2f2dc43
Worm.Win32.Fujack.dg - 75fbfe7a92bf11b59e3e6616b4cfc8db	Trojan-Spy.Pophot.hbn - 760dfe7eb26db590eeb0f54b7340e2f9
3rd malware set	
Trojan.Banker.afhd - 760f3a3f7520378278b84a25dd79dcd7	Backdoor.Sinowal.eed - 760f71e67ef40f75c8de084559f4a807
Packed.Tdss.i - 7628ab6d1aa42671f29efde798ac1225	Trojan-Spy.Zbot.ack - 76348e3e8016ce0663635ad6f7b8cf0e

Table 1. 3 groups of 4 malware samples with MD5 sums used in Tests 2 and 3

CAmp Name	**Log File Labels**	
	$\mathbf{TP_N}$	$\mathbf{TP_A}$
Kaspersky Internet Security 2010	Detected	Disinfected, Quarantined, Deleted
ESET Smart Security V4.2	Detected	cleaned by deleting - quarantined cleaned by deleting, cleaned
ZoneAlarm Extreme Security V9.1	File Repaired Failed	File Repaired Failed (Quarantined) Infected, File Repaired
BitDefender Total Security 2010	fail disinfect	deleted, disinfected, quarantined

Table 2. CAmps & labels used in Tests 1-3

were used in test 1. Tests 2 and 3 used three groups of four malware samples each, listed in Table 1. The malware were chosen and grouped based on their ability to execute harmoniously together. Current CAmp evaluations [1] adopt a simplistic and non-realistic approach of infecting a system with one malware sample per test. Current malware such as bots, backdoors and malicious downloaders initially infect a system with one malware which may then download and/or install several other malware. This produces a compromised system with several malicious objects consisting of many different malware types. We infected our system with four malware samples per test to evaluate a CAmp's effectiveness and resilience in a strongly compromised system containing a high number of malicious objects. This approach creates a realistic infected environment closely emulating current malware trends.

Estimating the size of S. As previously stated, all three tests were designed so that the input S consists entirely of malicious objects. Thereby $FP = 0$, $TN = 0$ and $S = TP + FN$. In test 1, S consists of a folder with 974 malware samples so its size is clearly 974. In all tests a clean copy of Windows XP-SP2 was installed in VMWare workstation and saved in a snapshot. Each tested CAmp was installed individually in the clean state snapshot. Once installed, the CAmp was fully updated and requested to perform a complete system scan. All four CAmps did not detect any malicious objects in the clean state. At this point in tests 2 and 3 the only changes made to the state was by execution of one group of four known malware samples. Any objects detected as malicious by a CAmp after executing a group of malware was assumed to be a TP of the specific CAmp.

Estimating the value of TP_A, TP_O, TP_N and FN. A CAmp's log file will minimally contain three pieces of information for every detected malicious object: location, malware name, and treatment type. For each tested CAmp, post-analysis of the treatment type labels listed in the log file, see Table 2, determined the output for TP_A and TP_N. We verified that the labels used for TP_A actually perform the label suggested treatment on malicious objects. This was done by attempting to locate, access and execute malicious .EXE files which appeared with a TP_A label in a CAmp's log file. For example, Kaspersky has three labels for TP_A: Disinfected, Quarantined, Deleted. For each of these labels we identified the corresponding files. The files labeled Quarantined were not allowed to execute via a double click. The malicious objects labeled Disinfected were scanned by two different CAmps and were reported benign. The malicious objects labeled Deleted were not found at the path listed in the log file. The labels listed under TP_N were found to leave the malicious object untreated, active and accessible to the system and by the user. For example, Kaspersky has one label for TP_N: Detected. The files with this label were found at the file path listed in the log file and we were able to freely access and execute the object without Kaspersky stopping the execution. These tests were conducted to determine which labels to use for TP_A and TP_N for all four CAmps and the results were confirmed as above in each case. In cases when a TP_O occurred we chose the best treatment option offered in the preference order of: Deletion, Disinfection, Quarantine. TP_O was calculated by manually counting the total number of times a CAmp requested a user to choose a treatment type for a detected malicious object. FN was estimated differently for each test as discussed below.

Result Tables. Individual test results are listed in Tables 3, 4, and 5. Each table presents the outputs TP, FN, TP_A, TP_O, and TP_N along with the TP, TP_A, and FN rates. Table 6 present the output average for each CAmp per test and overall. The three columns: Test 1, Test 2, and Test 3 include the TP and TP_A averages of each CAmp per test. For example, Kaspersky has a TP rate of 87.0% for Test 3, this was calculated by averaging the three malware set TP rates: 90.0%, 99.4%, 71.8% listed in Table 5. The final column is the overall TP and TP_A rates for each CAmp across all three tests. For example, Kaspersky overall TP_A rate is 93.3%, this was calculated by averaging the TP_A rates of each test: 93.6%, 100%, 86% listed in the same Table 6.

Test 1. The first test evaluates a CAmp's effectiveness in detecting and treating a folder containing a set of known malware samples. The test requests the CAmp to scan and perform treatment on any detected malicious objects found in the folder. This is considered a static file scan where none of the malware samples are executed. Using equation (2), we construct $A(S)$ for test 1 as S is a folder containing a set of known malware samples as follows:

$$A(S) = (FN, TP_A, TP_O, TP_N) \qquad (5)$$

To calculate FN, we totaled the number of missing filenames from our set of known malware samples in the log file of each CAmp. The results for test 1 are listed in Table 3 with $S = 974$. Only considering the TP rate, all four CAmps had very effective detection with BitDefender performing the best with 97.6%. Based on the TP_A rate, the results were mixed with Kaspersky achieving better detection and treatment over the other CAmps with 93.6%. Surprisingly Bitdefender had the worst TP_A rate of 67%, so it had very effective detection but not treatment. The TP_O output was varied with ESET and BitDefender producing very high values which also were the only two CAmps to produce TP_N. Table 3 suggests all four CAmps possessed highly effective detection in scanning a folder of known malware samples but some fall short of effective treatment. In this test, Kaspersky performed best with the highest TP_A rate, no TP_N, and minimal TP_O. The FN produced by all four CAmps should be reduced.

Test 2. The second test evaluates a CAmp's ability to perform detection and treatment in a system with an infected state and an unknown number of malicious objects where the CAmp is first installed and enabled in a clean state. After successful installation, the CAmp is left running with its detection and treatment abilities enabled. Four known malware samples are executed for a period of three minutes. At this point the CAmp is requested to perform detection and treatment on the complete system. Our construction of $A()$ for test 2 is the same as test 1 listed in equation (5) with S as a system in an infected state with an unknown amount of malicious objects. This amount is unknown since the original four samples may have infected additional files and processes, as well as possibly downloaded additional malware from the Internet. For this test, we assumed $S \geq 4$, since at least four malware samples are present by design in each set. $S = 4$ in cases where $TP + FN \leq 4$. If $TP + FN = N$ and $N > 4$, then $S = N$. To calculate FN, we compared the filenames of the four malware samples of each malware set to the log file of each CAmp and totaled the number of missing filenames in the log file, therefore $FN \leq 4$. The results for test 2 are listed in Table 4. The four CAmps produced the highest detection and treatment effectiveness overall in this test with almost every $TP_A = 100\%$ along with minimal FN, TP_O and no TP_N. In almost every instance just attempting to double click the malware sample file invoked the CAmp to perform detection and treatment not allowing the malware to ever execute. ESET in the 2nd malware set was the only instance with $S > 4$. Its log file contained 12 detected malicious objects, the four original malware samples and eight new malicious objects. In this case, $FN = 0$ since all four malware sample filenames appeared in the log file. We conjecture the eight additional TP occurred because one of the four malware samples may have executed and infected the system prior to detection and treatment. In the three instances where $FN > 0$, the undetected samples were executed with a double click. Newly infected objects were not detected by the CAmp.

Test 3. The third test evaluates a CAmp's ability to be successfully installed and perform detection and treatment in a system with an infected state and an unknown amount of malicious objects. The test starts by executing four known malware samples in a system with a clean state. After three minutes of execution, with the system now in an infected state, a CAmp is installed and attempts detection and treatment of malicious objects on the complete system. Our construction of $A()$ for test 3 including S is the same as test 2. S, TP and FN are reasonable estimates of the actual amount of malicious objects found on the system. S is different for each CAmp and malware set in test 3

CAmp Name	**TP**	**FN**	**TP_A**	**TP_O**	**TP_N**	**TP Rate**	**TP_A Rate**	**FN Rate**
Kasperskey	921	53	912	9	0	94.6%	93.6%	5.4%
ESET	936	38	814	118	4	96.1%	83.5%	3.9%
ZoneAlarm	920	54	896	24	0	94.5%	91.9%	5.5%
BitDefender	951	23	653	288	10	97.6%	67.0%	2.4%

Table 3. Test 1 results, S = 974

1st malware set									
CAmp Name	**TP**	**FN**	**TP_A**	**TP_O**	**TP_N**	**S**	**TP Rate**	**TP_A Rate**	**FN Rate**
Kasperskey	4	0	4	0	0	4	100%	100%	0%
ESET	4	0	4	0	0	4	100%	100%	0%
ZoneAlarm	2	2	2	0	0	4	50.0%	50.0%	50.0%
BitDefender	4	0	4	0	0	4	100%	100%	0%
2nd malware set									
Kasperskey	4	0	4	0	0	4	100%	100%	0%
ESET	12	0	10	2	0	12	100%	83.3%	0%
ZoneAlarm	4	0	3	1	0	4	100%	75.0%	0%
BitDefender	4	0	4	0	0	4	100%	100%	0%
3rd malware set									
Kasperskey	4	0	4	0	0	4	100%	100%	0%
ESET	3	1	3	0	0	4	75.0%	75.0%	25.0%
ZoneAlarm	3	1	2	1	0	4	75.0%	50.0%	25.0%
BitDefender	4	0	4	0	0	4	100%	100%	0%

Table 4. Test 2 results

1st malware set									
CAmp Name	**TP**	**FN**	**TP_A**	**TP_O**	**TP_N**	**S**	**TP Rate**	**TP_A Rate**	**FN Rate**
Kasperskey	199	22	198	1	0	221	90.0%	89.5%	10.0%
ESET	692	7	682	8	2	699	99.0%	97.5%	1.0%
ZoneAlarm	51	31	26	22	3	82	62.2%	31.7%	37.8%
BitDefender	241	23	193	42	6	264	91.3%	73.1%	8.7%
2nd malware set									
Kasperskey	1617	10	1614	3	0	1627	99.4%	99.2%	0.6%
ESET	19	15	19	0	0	34	55.9%	55.8%	44.1%
ZoneAlarm	16	13	3	11	2	29	55.2%	10.3%	44.8%
BitDefender	915	10	900	15	0	925	98.9%	97.2%	1.1%
3rd malware set									
Kasperskey	33	13	32	1	0	46	71.8%	69.5%	28.2%
ESET	6	12	2	3	1	18	33.3%	11.1%	66.7%
ZoneAlarm	1	14	0	1	0	15	6.7%	0%	93.3%
BitDefender	85	14	42	41	2	99	85.9%	42.4%	14.1%

Table 5. Test 3 results

	Test 1		**Test 2**		**Test 3**		**Overall**	
CAmp Name	**TP**	**TP_A**	**TP**	**TP_A**	**TP**	**TP_A**	**TP**	**TP_A**
Kasperskey	94.6%	93.6%	100%	100%	87.0%	86.0%	93.8%	93.3%
ESET	96.1%	83.5%	91.6%	86.1%	75.5%	74.2%	87.7%	81.2%
ZoneAlarm	94.5%	91.9%	75.0%	58.3%	41.3%	14.0%	70.2%	54.7%
BitDefender	97.6%	67.0%	100%	100%	92.0%	70.8%	96.5%	79.2%

Table 6. Overall TP and TP_A result Tests 1-3

for two reasons. First, it is unknown how many malicious objects exist in the infected state after executing the four malware samples. Second, each CAmp has different detection capabilities which result in different amounts of detected malicious objects documented in their log files. FN was calculated by first submitting the malware samples to the binary analysis platforms CWSandbox [19] and Anubis [4] for execution analysis. These two platforms execute malware samples and extensively record and report the execution behavior. We compiled a list of .EXE files appearing in both reports that were either the original malware file, a file created by the malware or a running process created or modified by the malware. By only using .EXE files appearing in both reports, we can reasonably assume these are essential files which are created and/or modified every time the malware is executed. The lists of .EXE files for each malware sample was compared to each CAmp's log file for filename matching. The total number of .EXE filenames not found in a CAmp's log file was assumed not detected by that specific CAmp and counted as a false negative (FN). The results for test 3 are listed in Table 5. The results of each malware set varied greatly for the tested CAmps with malware set 3 producing the worst results. In the first malware set, ESET achieved the highest and ZoneAlarm the lowest TP and TP_A rates. In the second malware set, Kaspersky performed best while ESET performed the worst. In the third set, BitDefender achieved the most effective detection with a TP rate of 85.9% and Kaspersky achieved the best detection and treatment with a TP_A rate of 69.5%. In all three sets, ZoneAlarm had the least effective detection and treatment primarily becuse of its high FN and TP_O rates. The 0% TP_A rate in the third malware set implies ZoneAlarm found it difficult to detect and automatically treat infected objects when installed and executed in an infected state. All four CAmps produced FN and TP_O in all three malware sets except ESET with no TP_O in the second malware set. TP_N was minimal in all three malware sets for all four CAmps.

Discussion. The overall average results of the four CAmps for tests 1-3 are listed in Table 6. The four CAmps performed best in test 2 followed by test 1 and test 3. It seems a CAmp may have the hardest time performing effective detection and treatment when installed in an infected state. With an overall TP_A rate of 93.3%, Kaspersky had the most effective detection and treatment of all four CAmps across all three tests. All the tests produced FN and TP_O, with TP_N occuring in only two tests. The malicous objects pertaining to these three outputs remain active on a system. CAmps need to improve detection and treatment to minimize this amount of active threats. Based on the TP_A rate in Table 6, all four CAmps produced inconsistent detection and treatment. This implies a CAmp's detection and treatment capabilities may be impaired based on the scenario in which they are requested to perfom. This was particularly evident in test 3, where the CAmps were installed and executed in an infected state. Also in test 3, the four CAmp's TP_A rate per malware set were very inconsistent. This indicates a CAmp's capabilities can possibly be impaired by the type of malware and number of malicious objects executing in the system. We originally tested six CAmps and purposely did not report the results of two CAmps. In a majority of our tests the CAmp AVG would either not fully run a detection scan properly, not install correctly or for unknown reasons cause a system shutdown. This occured mostly in tests 2 and 3. The CAmp G-Data only produced TP_O and FN in every test but its detecton was highly effective. We could not explain why G-Data never produced one instance of TP_A. We decided not to include the output concluding the cause may be that automatic treatment was disabled in the trial version used in testing and not some fundamental logic of the CAmp.

5 Limitations

All tests were carried out in a virtual environment which forcibly disallowed the use of VM-aware malware. The malware executed during our tests were strictly Win32 malware samples. Malware with an execution method that is not a mouse double click were not used. We are currently creating tools allowing accurate tracking of a malware infection in a virtual machine. These tools will produce an accurate count instead of an estimate of malicious objects present in a system. We are also considering methods to estimate an error rate of our current measures of S. Choosing a single malware upload date may reduce diversity and create unintended bias in testing. We are currently testing with several different dates to reduce any possible bias.

6 Conclusions and Future Work

In this paper, we have presented a new approach to evaluate the detection and treatment effectiveness of a commercial anti-malware program (CAmp) against malicious objects with a standardized output. We partition the standard output of true positive, TP, into three groups: TP_A, TP_O and TP_N. This partitioning incorporates detection and treatment effectiveness by reporting which malicious objects were detected and treated automatically (TP_A), detected and treated with a user option (TP_O) and detected and not treated (TP_N). These new outputs along with FN provide a more accurate and realistic evaluation of CAmp effectiveness than just (TP, FN). Using these outputs we evaluated four CAmps in three tests representing realistic scenarios of a user dealing with malware on a system. Our results reveal TP_A rates lower than TP rates. This implies detection and treatment was, in many test, less effective than detection alone. The tests also reveal many CAmps use misleading labels in their log files which result in several instances of FN, TP_O and TP_N. The malicious objects belonging to these outputs were verified by us to remain active and accessible on the system. Our test results suggest an accurate and realistic evaluation of a CAmp's effectiveness against malicious objects must equally consider detection (TP, FN) and treatment (TP_A, TP_O, TP_N) instead of just detection alone. Based on our results we conclude a CAmp is highly effective in the detection and treatment of malicious objects when it produces a high TP_A rate and a low FN rate. Our future work includes ongoing testing of a broad corpus of CAmps and malware along with a more accurate measure of the total amount of malicious objects present in an infected system. We will also extend our evaluation methodology to include false positives and true negatives to evaluate a CAmp's effectiveness in dealing with benign objects.

Acknowledgments. This work is partially supported by grants from AFOSR, ONR, AFOSR MURI, and the State of Texas Emerging Technology Fund.

References

[1] http://www.anti-malware-test.com.

[2] The fundamental principles of testing. Technical report, AntiMalware Testing Standards Organization, 2008.

[3] Whole product testing guidelines. Technical report, AntiMalware Testing Standards Organization, 2010.

[4] http://anubis.iseclab.org/.

[5] M. Christodorescu and S. Jha. Testing malware detectors. *ISSTA '04: Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis*, pages 34–44, 2004. ACM Press, New York, NY, USA. http://doi.acm.org/10.1145/1007512.1007518.

[6] M. Christodorescu, S. Jha, D. Maughan, D. Song, and C. Wang, editors. *Malware Detection.* Springer, 2007.

[7] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia. A behavioral approach to worm detection. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 43–53, New York, NY, USA, 2004. ACM Press.

[8] E. Filiol. *Computer Viruses: from Theory to Applications.* IRIS International series, Springer Verlag, 2005. ISBN 2-287-23939-1.

[9] S. Gordon and R. Ford. Real world anti-virus product reviews and evaluations - the current state of affairs. In *Proceedings of the 1996 National Information Systems Security Conference*, 1996.

[10] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium (Security'08)*, 2008.

[11] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang. Effective and efficient malware detection at the end host. In *18th Usenix Security Symposium*, 2009.

[12] A. Marx. Guideline to anti-malware software testing. In *European Institute for Computer Anti-Virus Research (EICAR) 2000 Best Paper Proceedings*, 2000. pp.218-253.

[13] J. A. Morales, P. J. Clarke, Y. Deng, and B. G. Kibria. Identification of file infecting viruses through detection of self-reference replication. *Journal in Computer Virology Special EICAR conference invited paper issue*, 2008.

[14] J. A. Morales, P. J. Clarke, B. Kibria, and Y. Deng. Testing and evaluating virus detectors for handheld devices. *Journal in Computer Virology*, September 2006.

[15] R. Moskovitch, Y. Elovici, and L. Rokach. Detection of unknown computer worms based on behavioral classification of the host. *Comput. Stat. Data Anal.*, 52(9):4544–4566, 2008.

[16] I. Muttik and J. Vignoles. Rebuilding anti-malware testing for the future. In *Virus Bulletin Conference*, 2008.

[17] J. C. Rabek, R. I. Khazan, S. M. Lewandowski, and R. K. Cunningham. Detection of injected, dynamically generated, and obfuscated malicious code. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 76–82, New York, NY, USA, 2003. ACM Press.

[18] http://en.wikipedia.org/wiki/Receiver-operator_characteristic.

[19] http://www.sunbeltsoftware.com/Malware-Research-Analysis-Tools/Sunbelt-CWSandbox/.

[20] P. Szor. *The Art of Computer Virus Research and Defense.* Symantec Press and Addison-Wesley, 2005. ISBN 9-780321-304544.

[21] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda. Panorama: capturing system-wide information flow for malware detection and analysis. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 116–127, New York, NY, USA, 2007. ACM.