IPSJ Transactions on Advanced Computing Systems Vol.7 No.1 14–22 (Mar. 2014)

Regular Paper

Screening Legitimate and Fake/Crude Antivirus Software

Masaki Kasuya[1,a) Kenji Kono[1,2,b)

Received: July 25, 2013, Accepted: December 4, 2013

Abstract: Fake antivirus (AV) software, a kind of malware, pretends to be a legitimate AV product and frightens computer users by showing fake security alerts, as if their computers were infected with malware. In addition, fake AV urges users to purchase a "commercial" version of the fake AV. In this paper, we search for an *indicator* that captures behavioral differences in legitimate AV and fake AV. The key insight behind our approach is that legitimate AV behaves *differently* in clean and infected environments, whereas fake AV behaves *similarly* in both environments, because it does not analyze malware in the infected environments. We have investigated three potential indicators, file access pattern, CPU usage, and memory usage, and found that memory usage is an effective indicator to distinguish legitimate AV from fake AV. In an experiment, this indicator identifies all fake AV samples (39 out of 39) as fake and all legitimate AV products (8 out of 8) as legitimate. It is impractical for fake AV to evade this indicator because to do so would require it to detect malware infections, just as legitimate AV does.

Keywords: antivirus software, fake antivirus software, behavior analysis, malware

1. Introduction

Fake antivirus (AV) software is a severe threat to our computer systems [24]. It pretends to be actual AV software and shows false security warnings to users as if their computer systems were infected with malicious software (malware) [23]. Fake AV persuades victim users to purchase a useless, "commercial" version of the fake AV to eliminate bogus threats. "Crude AV" poses a similar threat to fake AV. It differs from fake AV in that it detects malware, but its detection quality is too low to be practical. Fake/crude AV is sometimes used to collect sensitive information such as credit card numbers [9]. For example, AntiVirus 2009, a fake AV sample, collects credit card numbers and e-mail addresses of the victims.

The threat of fake AV is real. Symantec detected 43 million installation attempts of fake AV from July 2008 to Jun 2009 [9]. According to Rajab et al. [22], fake AV accounts for 15% of all malware detected by Google's malware detection infrastructure [21]. Fake AV business earns tremendous revenue. Stone-Gross et al. revealed that three kinds of fake AV have earned more than $130 million dollars [25]. McAfee disclosed that the annual revenue of one vendor of fake AV exceeded $180 million dollars [20]. To distribute fake AV, software download sites are sometimes exploited. In fact, one famous download site, CNET, distributed a fake AV sample called RegGenie in 2012 [7].

Fake/crude AV is similar to a social engineering attack [26]; the victim users are deceived and never suspect that fake/crude AV is not legitimate because it is carefully designed to look like legitimate AV. One approach for defending against fake/crude AV is to use signature-based approaches. However, signature-based approaches are exploit-specific, and a signature must be prepared for each instance of fake/crude AV. Therefore, these approaches cannot detect previously unseen instances of fake/crude AV [18].

In this paper, we reveal behavioral differences between legitimate and fake/crude AV, and propose an *indicator* that captures the behavioral differences in AV software. The key insight behind our approach is that legitimate AV behaves differently in 1) clean environments and 2) infected environments, while fake/crude AV would not show such differences. A clean environment is the one in which no malware has been installed, whereas an infected environment is the one in which malware has been installed. Fake/crude AV is not expected to show behavioral differences in clean and infected environments because it does not analyze malware samples in the infected environment. On the other hand, legitimate AV instances are expected to show the differences because they deeply analyze suspicious instances in the infected environments.

Our indicator can be used in software download sites such as CNET [1] and PCMAG [3]. When AV samples are uploaded to download sites with the tag indicating AV, they can be checked as to whether they are legitimate or fake/crude. Since our indicator works by capturing behavioral differences instead of using signature, it can detect the latest fake/crude AV instances.

This paper shows that "memory usage" is an effective indicator of fake/crude AV software. Surprisingly, crude AV does not show differences in memory usage between clean and infected environments. In our approach, the memory usages of AV-like software in clean and infected environments are compared statistically. The Levene Test [15], an inferential statistic, is used to

1 Department of Information and Computer Science, Keio University, Yokohama, Kanagawa 223–8522, Japan
2 CREST, Japan Science and Technology Agency
a) kasuya@sslab.ics.keio.ac.jp
b) kono@ics.keio.ac.jp

A preliminary version of this paper will appear in The Seventh International Conference on Emerging Security Information, Systems and Technologies.

assess the equality of variances in memory usage. If a software sample that is suspected to be fake/crude AV has statistically the same distribution of memory usage in both environments, it is considered fake. Otherwise, it is considered legitimate.

Since our indicator is based on behavioral differences, fake/crude AV has to mimic the behaviors of legitimate AV in order to evade it. It is not easy to mimic behaviors of legitimate AV. If fake/crude AV samples change their memory consumption at random, our approach can detect fake/crude AV correctly because it compares memory consumption under several settings, i.e., clean/clean, infected/infected, and clean/infected.

To demonstrate the usefulness of our approach, we have conducted experiments on 39 "real" fake/crude AV samples and 8 legitimate AV products. The results show that our indicator can identify all 39 fake/crude samples, which means there are no false negatives, and all 8 legitimate products, which means there are no false positives.

The remainder of this paper is organized as follows. Section 2 describes the differences between fake AV and crude AV, and the current criteria to distinguish them from legitimate AV. Section 3 explains our basic approach and shows how to distinguish fake/crude AV from legitimate AV by using the Levene Test. Section 4 presents our experimental results. Discussion and related work are presented in Sections 5 and 6. Section 7 concludes the paper.

2. Fake AV and Crude AV

There are two types of malicious AV software: fake AV and crude AV. To understand the difficulties of distinguishing between fake/crude AV and legitimate AV, this section describes the behaviors of fake and crude AV, and briefly introduces recent guidelines to distinguish fake/crude AV and legitimate AV.

2.1 Fake AV

Fake AV mimics the behavior of legitimate AV and shows bogus security warnings without scanning for malware infections in the victims' file systems. To make the behavior resemble that of legitimate AV, fake AV searches the file system to obtain file and/or directory names to be displayed in warning messages. For example, Security Antivirus [13], a fake AV, displays the following message:

Virus name:
Virus.Win32.Faker.a
Infected file:
C:\Documents and Settings\Kasuya\Recent\snl2w.dll
Description:
These programs steal MSN Messenger passwords...

Pathname, `C:\Documents and ... snl2w.dll`, is the real one in the victim's file system. By showing real pathnames in the victim's file system, the fake AV deceives victims into believing the machine is infected with malware and encourages them to the purchase a product version of the fake AV.

The directory traverse of fake AV makes it difficult to distinguish it from legitimate AV. When observed from the outside, fake AV traverses directories just as legitimate AV does. Security Antivirus traverses most directories that all legitimate AV products commonly access. According to our investigation, the access coverage of Security Antivirus is over 99.7% (= 2,393 / 2,400). This result means that Security Antivirus carefully takes access patterns of legitimate AV into account. In other words, we cannot use directory traversal as an indicator to distinguish fake AV from legitimate AV.

2.2 Crude AV

Crude AV is low-quality AV software whose detection accuracy is too low to be useful. Crude AV differs from fake AV in that it scans file systems for malware and detects the infection. At the same time, crude AV differs from legitimate AV in that it cannot detect a large portion of widely deployed malware. To confirm that the detection rate of crude AV is very low, we measured the detection rate of Anti-Virus Elite [14], a well-known crude AV. We installed 905 unique instances of malware in Windows XP SP3. Anti-Virus Elite detected only 74 samples. The detection rate was 8.2%. Kaspersky, an example of legitimate AV, detected all 905 samples, i.e., its detection rate was 100.0%.

Crude AV is usually classified into malware. According to VirusTotal [4], an online antivirus scan service, Anti-Virus Elite is classified as malware in 65% (28 out of 43) of commercial AV products. Crude AV is malware because there are sites that urge the visitors to buy a "product" version, which in most cases is just as poor as the crude AV.

Crude AV blurs the boundary between fake and legitimate AV, and makes it more difficult to distinguish fake/crude AV from legitimate AV. Crude AV traverses file systems and inspects suspicious files that may contain malware. Aside from the quality of detection, crude AV behaves very similarly to legitimate AV.

2.3 Current Criteria to Distinguish Legitimate and Fake/Crude AV

Recently, a security industry has published a white paper [12] to help end-users identify fake security products such as fake AV. The document provides a helpful checklist to judge whether the users' computers are infected with fake AV. The checklist says, for instance, that fake AV reports an unreasonably high number of infections, shows a popup window frequently that warns your machine is infected with malware, and suggests the purchase of a commercial version.

This checklist is useful for manual inspection for discovering fake AV. However, these criteria do not suit automated distinction of fake/crude and legitimate AV. Suppose that we attempt to automate the process of counting the number of reported infections. Since the reports are shown in natural language, it is not easy for computers to understand the reports. Even if we could interpret the reports, there are samples of fake AV that do not show up in a lot of reports. For example, Anti Spyware Expert has only 18 reports of infection.

Furthermore, this checklist does not address how to distinguish crude AV from legitimate AV. Since crude AV behaves similarly to legitimate AV aside from the quality of detection, it is almost impossible to draw up a guideline for crude AV.

3. Searching for Indicators

In this section, we describe our search for fake/crude AV indicators. As mentioned above, the key insight behind our approach is that legitimate AV behaves differently in clean and infected environments while fake/crude AV behaves similarly in both environments. A good indicator captures behavioral differences only in legitimate AV.

3.1 What is a Good Indicator?

A fake/crude AV indicator should satisfy at least three requirements. First, it should be applicable to as many instances of fake/crude AV as possible. In particular, it should be applicable to previously-unseen instances of fake/crude AV.

Second, it should be impractical for fake/crude AV to evade the indicator. The criminals that make use of fake/crude AV for their own profit do not want to spend a lot of money on development because it would reduce their revenues. A fake/crude AV indicator would thus be ideal since criminals would have to incorporate the same functionalities as legitimate AV in order to evade it and they would cost a fortune to develop software equivalent to legitimate AV.

Third, a fake/crude AV indicator should enable automatic distinction of fake/crude AV from legitimate AV. If the distinction process is automated, it can be incorporated into legitimate AV or deployed on software download sites [1], [3]. The guidelines presented in Section 2.3 assume manual inspection of several aspects of suspicious behavior of AV-like software. In this paper, we seek a fake/crude AV indicator that does not require manual intervention. Rather than relying on visual inspection, we seek a fake/crude AV indicator from information that can be obtained in a systematic way. For example, we look for system call patterns or resource usage patterns that differentiate fake/crude AV from legitimate AV.

One approach to deriving a fake/crude AV indicator is to use binary analysis. We cannot rely on the source code because the source code of malware is not available in public. Binary analysis has the potential to identify a lot of operations (e.g., system calls and their arguments) that malware may do. However, it is not easy to apply binary analysis to fake/crude AV because malware can be obfuscated using a technique like binary obfuscation [27]. In addition, it is not straightforward to find a sequence of operations that can differentiate fake/crude AV from legitimate AV. Hence, we decide not to take this approach.

3.2 Basic Approach

Our basic approach is to *compare* potential indicators obtained in *clean* and *infected* environments. A clean environment is one in which no malware has been installed. We assume that an execution environment just after installing an operating system from read-only media such as DVD-ROM is clean. Therefore, an environment with harmless files is also clean. A clean environment can be prepared by using the recent technology of virtual machines. Once a clean environment has been prepared, we can reuse it by saving it as a virtual machine image. An infected environment is one in which malware has been installed. This environment is also saved as a virtual machine image for reuse.

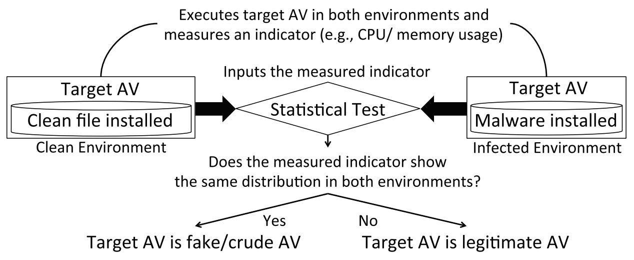

Executes target AV in both environments and measures an indicator (e.g., CPU/ memory usage)
Inputs the measured indicator
Target AV
Clean file installed
Clean Environment
Statistical Test
Target AV
Malware installed
Infected Environment
Does the measured indicator show the same distribution in both environments?
Yes
No
Target AV is fake/crude AV
Target AV is legitimate AV

Fig. 1 Our basic approach.

Figure 1 illustrates the basic approach used in this paper. For each sample of AV-like software (it is unknown whether the sample is fake/crude AV or legitimate AV at this point of time), indicators are measured in clean and infected environments and *statistically* compared. The key insight behind this approach is that legitimate AV behaves *differently* in clean and infected environments while fake/crude AV behaves *similarly* in clean and infected environments, because legitimate AV thoroughly analyzes files suspected to be infected with malware. fake/crude AV *cannot* change its behavior depending on the presence of an infection because it does not detect malware infection.

This approach satisfies the three requirements described in Section 3.1. First, the fake/crude AV indicator does not use features specific to each instance of fake/crude AV, and thus, should be applicable to a wide variety of fake/crude AV. As shown in Section 4, our indicator successfully identified all (39 out of 39) fake/crude AV samples and all (8 out of 8) legitimate AV samples. Second, it is impractical to try to evade the indicator. Since fake/crude AV must change its behavior depending on the presence of malware infection, it must be equipped with the detection facilities that are equivalent to legitimate AV; that is, criminals must develop a legitimate AV to evade the indicator. Finally, the distinction process can be automated. There is no need for manual intervention since the indicator can be used in a systematic way and indicator measurements are compared using a statistical test.

3.3 Examining Potential Indicators

To discover a good indicator, we examine three candidates that can be obtained systematically: 1) the file access pattern, 2) CPU usage, and 3) memory usage. To obtain them, we install hooks to get the file accesses and Performance Monitor, a default application of the Windows OS, to get the CPU usage and memory usage.

3.3.1 File Access Pattern

While a legitimate AV has to investigate a file's content to determine whether it is infected with malware, fake/crude AV only traverses directories to obtain real pathnames; it does not access files as often as legitimate AV does, because fake/crude AV does not look hard for malware infection.

Unfortunately, file access patterns are not a good indicator of fake/crude AV because the patterns of some fake/crude AV samples are similar to those of legitimate AVs. **Figure 2** shows the similarity of file access patterns between 8 legitimate AV products and 39 fake/crude AV samples. Each bar corresponds to one

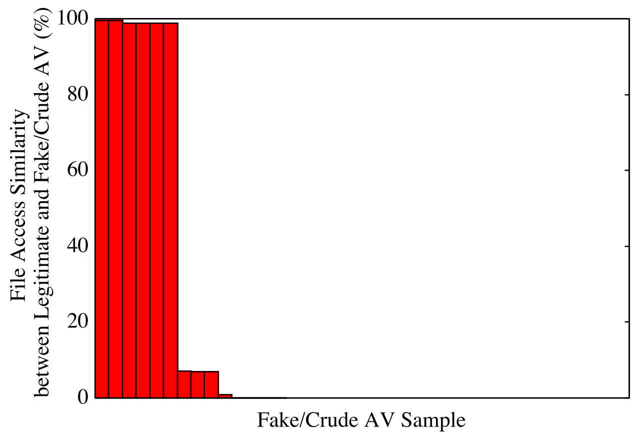


Fig. 2 Similarity of file access patterns between legitimate AV and fake/crude AV. The file access pattern is not a good indicator because 6 out of 39 fake/crude AV samples access files accessed by legitimate AV products.

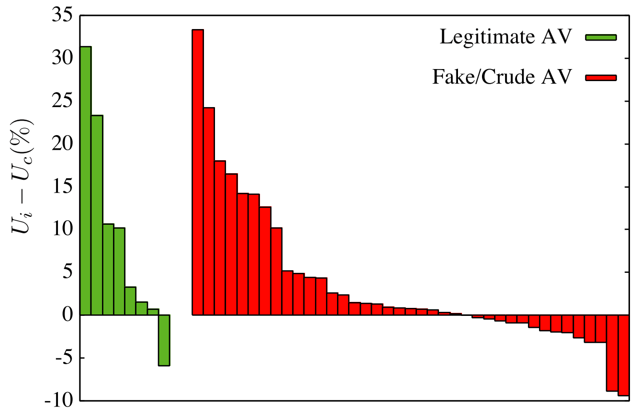


Fig. 3 Comparison of user-mode times in clean and infected environments. U_i and U_c represent user-mode time ratios in an infected and a clean environment, respectively. The green bars show $U_i - U_c$ of legitimate AV products, and the red bars show those of fake/crude AV samples. Regardless of whether it is fake/crude or legitimate, $U_i - U_c$ ranges from −10% to 30%.

fake/crude AV sample, and shows the ratio of files accessed by each fake/crude AV sample to those commonly accessed by legitimate AV samples. (There are 10 bars in the figure because the remaining 29 samples do not access the files). Figure 2 shows that six fake/crude AV samples resemble legitimate AV products in terms of file access.

3.3.2 CPU Usage

Next, we investigate CPU usage. Since a malware scan such as signature matching is executed in user mode not kernel mode, we focus on the proportion of user mode time of CPU usage. User-time is expected to increase in legitimate AV if it is executed in infected environments because sophisticated scanning algorithms consume a lot of user-mode time. On the other hand, fake/crude AV is not expected to increase user-mode time in infected environments because it does not search for malware infection.

In spite of our expectations, the differences in user-mode time between clean and infected environments are not useful for distinguishing fake/crude AV from legitimate AV. **Figure 3** shows the differences in user-mode time ratio between clean and infected environments. The y-axis shows $U_i - U_c$, where U_i stands for the user-mode time ratio measured in an infected environment and U_c stands for the user-mode time ratio measured in a clean environment. The green bars show $U_i - U_c$ of legitimate AV products, and the red bars show those of fake/crude AV samples. The overall trend in the user-mode time ratios is the same in the legitimate products and fake/crude samples. $U_i - U_c$ ranges from −10% to 30%.

3.3.3 Memory Usage

Finally, we examine memory usage. Memory usage is expected to increase in infected environments in legitimate AV because it requires more memory to perform in-depth analyses of malware. On the other hand, fake/crude AV is not expected to increase memory usage because it should behave similarly in clean and infected environments.

Our preliminary results show memory usage can be an effective indicator to distinguish legitimate AV from fake/crude AV. **Figure 4** shows the memory usage of one legitimate AV product, one fake AV sample and one crude AV sample. It reveals memory usage can differ in legitimate and fake/crude AV. In the legitimate AV product, the memory usage increases in infected environments. However, the fake/crude AV sample does not show such increase in memory usage in infected environments; it shows almost the same trend in both environments.

Figure 5 shows V_i/V_c for each legitimate AV product and fake/crude AV sample. V_i stands for the variance in an infected environment and V_c stands for the variance in a clean environment. As you can see from the figure, V_i/V_c shows different trends for legitimate AV and fake/crude AV. This suggests that memory usage is a good indicator of fake/crude AV.

Table 1 shows the summary of the compared features, monitoring cost of the features, and the effectiveness (results) of each potential indicator.

3.4 Memory Usage as an Indicator

This section describes a concrete method to distinguish fake/crude AV from legitimate AV. On the basis of examinations in the previous sections, we choose memory usage as an indicator of fake/crude AV. A sample of AV-like software is installed in clean and infected environments and the memory usage is measured in each environment. An infected environment is prepared by installing 905 unique instances of malware (about 500 MB in total), and a clean environment is prepared by installing 500 MB of clean files. The installed malware instances and clean files are the same size and have the same directory structure. Note that we do not have to prepare these environments every time a test is performed, because a virtual machine image can be copied for reuse.

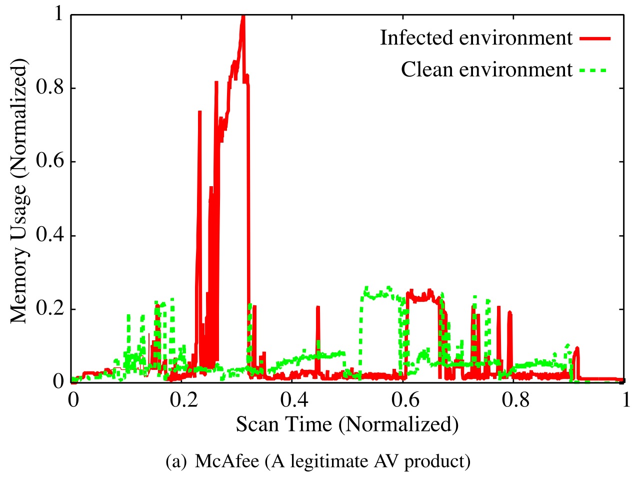
Memory usage is measured every second in each environment. For ease of mathematical formalization, the clean and infected environments are numbered 1 and 2. The measured memory usage values are grouped into a sequence for each environment and represented by Y_i, where i denotes the number of group ($1 \leq n \leq 2$).

If the distributions in Y_1 (clean env.) and Y_2 (infected env.) are not statistically different, we conclude that the tested sample of AV-like software is fake/crude AV. Otherwise, we conclude that the tested sample is legitimate AV.

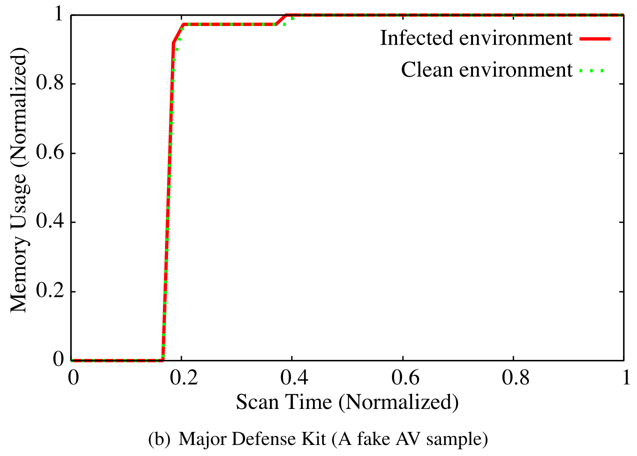
To compare the memory usage distributions in each environment, we use the *Levene Test* [15], a well-known inferential statistic used to assess the equality of variances in different samples. It

Table 1 Summary of our preliminary results.

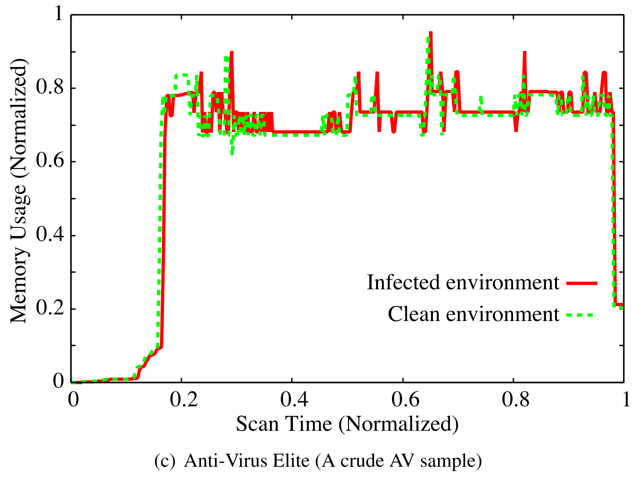
Indicators	Compared Features	Monitoring cost	Results
File access pattern	File access patterns	Hook open system call	bad
CPU usage	CPU usage in user mode	CPU usage obtained every second	bad
Memory usage	Variances of memory usages	Memory usage obtained every second	good



(a) McAfee (A legitimate AV product)



(b) Major Defense Kit (A fake AV sample)



(c) Anti-Virus Elite (A crude AV sample)

Fig. 4 Memory usages of legitimate AV, fake and crude AV. Legitimate AV significantly consumes a lot of memory when it detects malware. On the other hand, fake/crude AV hardly changes their usage in going from clean to infected environments. Memory usage and scan time are normalized.

tests the null hypothesis that the population variances are equal. The Levene Test compares the distributions of two sequences Y_i and Y_j. If the results of the test are less than the significance level (0.05 in this paper), the difference is statistically significant. In our method, memory usage is measured M times to mitigate fluctuations and the Levene Test is repeated. If the M results of the Levene Test are all less than 0.05, we consider that the distributions are different. Since it is time-consuming to measure indicators M times, we measure indicators $\lceil \sqrt{M} \rceil$ times and perform the Levene Test on any pair of the measured indicators.

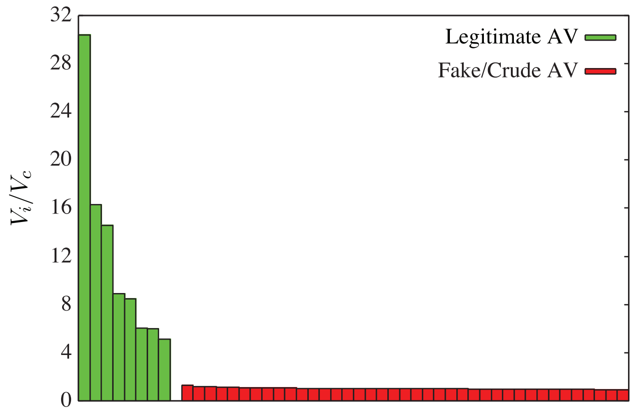


Fig. 5 Comparison of variances in clean and infected environments. V_i and V_c represent the variances of memory usage distributions in an infected and a clean environment, respectively. All legitimate AV products significantly increase the variances in the infected environment. However, fake/crude AV samples hardly change variances in these environments.

4. Experiments

This section shows that memory usage can be used to distinguish fake/crude AV from legitimate AV. We collected 8 legitimate AV products listed in **Table 2**. In addition to them, 39 fake/crude AV samples listed in **Table 3** from malware collection sites [2], [19] and Malware Domain List [17]. Clean and infected environments were prepared by using KVM with Qemu 1.1.1, in which Windows XP SP3 was installed and 1GB of memory is allocated. Although we used Windows XP in the experiments, our approach was not limited to a specific OS. Memory usage was obtained from Windows process structures (EPROCESS entries), because some fake/crude AVs interfere with the access to system information through Performance Monitor. These environments were prepared as described in Section 3.4. The Levene Test was repeated 9 times. In other words, M equaled 9. Since $\lceil \sqrt{M} \rceil$ was 3 in this case, memory usage in each environment was measured 3 times.

The Levene Test used memory usages obtained from different environments as shown in **Fig. 6**, and counts of less than significance level (0.05) were gathered. If the count reached 9, the tested AV sample was identified as legitimate. Otherwise, it was identified as fake/crude.

Our method correctly identified all 39 fake/crude AV samples and 8 legitimate AV products. The results are shown in **Table 4**. In the table, class means whether tested samples are legitimate or fake/crude. C and I mean memory usage in clean environment

Table 2 Legitimate AV products.

Avast Pro Antivirus 7.0.1426	G Data Antivirus 2011 21.1.0.1
AVG Antivirus 2012.0.1913	Kaspersky Anti-Virus 2011 11.0.2.556
McAfee VirusScan 15.0.294	ESET NOD32 Antivirus 4.2.71.2
Norton AntiVirus 18.7.0.13	Panda Antivirus Pro 2011 10.00.00

Table 3 Fake/Crude AV samples.

XP Internet Security 2011	XP Internet Security 2012 6.0.2900.2180
XP Home Security 2011	XP Home Security 2012 6.0.2900.2180
XP Anti Spyware 2011	XP Anti Spyware 2012 6.0.2900.2180
XP Antivirus 2011	XP Antivirus 2012 6.0.2900.2180
XP Security 2011	XP Security 2012 6.0.2900.2180
XP Total Security 2011	PC Privacy Cleaner 1.0.22.4
Patchup Plus	Virus Remover 2008 1.0.15.2
Security Tool	Virus Remover 2009 1.0.9.0
System Security	Anti Spy Safeguard 1.0.0.0
XL Guarder	Security Antivirus 2.0.2.18
Security Shield	Major Defense Kit 1.0.0.0
Protect Code	Anti Spyware Bot 9.6.9
Adware Bot 12.0.6	Security Defender 1.6.812.0
Reg Clean 1.0.0.1	Malware Removal Bot 12.0.6
Onescan 1.0.0.1	Anti Spyware Expert 1.0.22.2
Anti-Spyware 12.0.6	Anti-Virus Elite v5.0
Error Sweeper 2.8.0	Pest Detector 1.0.0.0
Registry Smart 2.10.0	Netcom3 PC Cleaner 9.1.10
Red Cross 1.0.0.0	Peak Protection 1.0.0.0
Privacy Control 2.6.0.0	

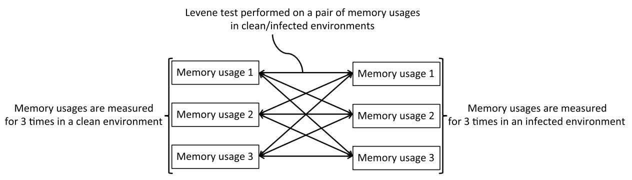

Levene test performed on a pair of memory usages in clean/infected environments
Memory usages are measured for 3 times in a clean environment
Memory usage 1
Memory usage 2
Memory usage 3
Memory usage 1
Memory usage 2
Memory usage 3
Memory usages are measured for 3 times in an infected environment

Fig. 6 Memory usage is measured 3 times in a clean and an infected environment, respectively (we obtain 3 memory usages for a clean environment and another 3 memory usages for an infected environment). The Levene test is performed on any pair of the memory usages. Each arrow means one Levene test is performed on the pair of the memory usage for a clean environment (lefthand side of the arrow) and the memory usage for an infected environment (righthand side of the arrow). Since we have 3 memory usages for a clean and an infected environment respectively, we perform 9 (= 3 * 3) Levene tests in total for all the pairs of the memory usages.

and infected environment, respectively, e.g., C_1 is the memory usage obtained from clean environment 1. When a pair shows statistical significance, the result of the Levene Test is shown as "✓" in Table 4. Otherwise, it is shown as "X" in the table. If a sample shows the count of "✓" is 9, the verdict is legitimate AV. Otherwise, the verdict is fake/crude AV.

In all legitimate AVs in Table 4 (upper 8 samples), the number of "✓" is 9. It means memory usages of legitimate AV between the infected and clean environments are always statistically significant. In other words, legitimate AV consumes a lot of memory to analyze and detect malware; i.e., the memory usage clearly increases in the infected environments. However, the memory usage does not increase in the clean environments because legitimate AV does not analyze malware. **Figure 7** and **Fig. 8** show the memory usages of McAfee and AVG. Compared with the memory usages in the clean environments, AVG and McAfee consume more memory in the infected environments. All the other legitimate AVs show a similar pattern.

In 19 fake/crude AV samples out of 39, the number of "✓" is zero. In the 29 samples out 39, the number of "✓" is less than

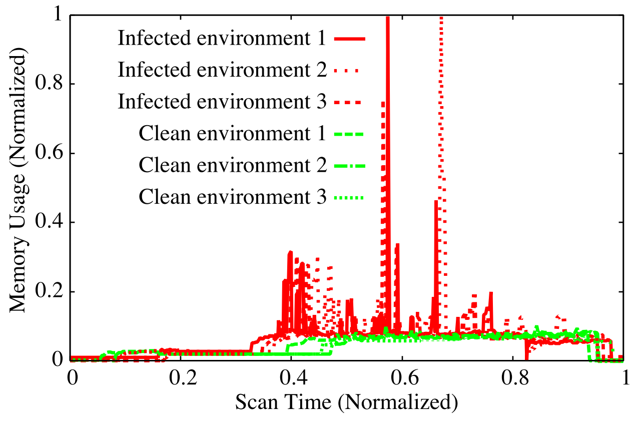


Fig. 7 Memory usages of AVG, a legitimate AV product. When it detects malware, the memory usages clearly increase.

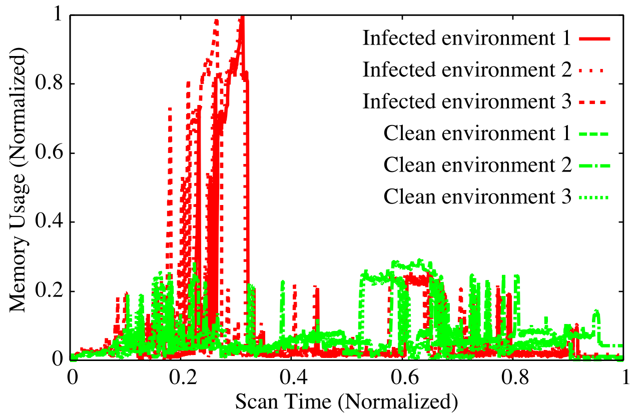


Fig. 8 Memory usages of McAfee, a legitimate AV product. It consumes a lot of memory in infected environment.

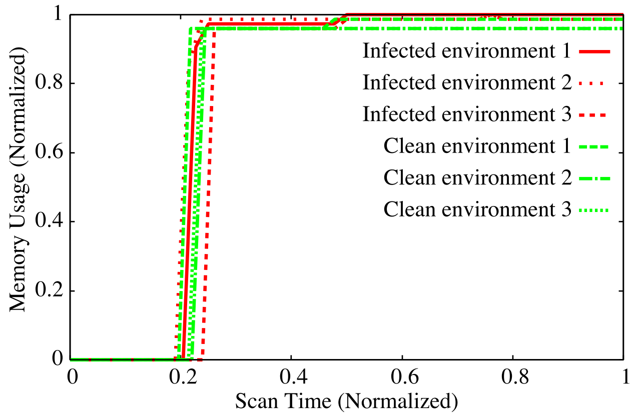


Fig. 9 Memory usage of Red Cross. The memory usages are almost the same between infected and clean environments. Memory usage and scan time are normalized.

or equal to 3. The result shows the memory usage of fake/crude AVs does not show statistical differences in the clean and infected environments. **Figure 9** shows RedCross's memory usage. The figure shows the memory usages between the clean and infected environments are almost the same. **Figure 10** shows Pest Detector's memory usage (the count of "✓" is 3). In Fig. 10, the memory usage in infected environment 3 is obviously different from the others. As a result, all pairs including the memory usage in infected environment 3 show statistical significance.

In two fake/crude AV samples, the number of "✓" is 6. **Figure 11** shows the memory usage of XP AntiVirus 2011. Com-

Table 4 Each Levene Test's result and the verdict.

Name	Class	C_1 & I_1	C_1 & I_2	C_1 & I_3	C_2 & I_1	C_2 & I_2	C_2 & I_3	C_3 & I_1	C_3 & I_2	C_3 & I_3	Total of ✓	Verdict
Avast	Legitimate	✓	✓	✓	✓	✓	✓	✓	✓	✓	9	Legitimate
AVG	Legitimate	✓	✓	✓	✓	✓	✓	✓	✓	✓	9	Legitimate
McAfee	Legitimate	✓	✓	✓	✓	✓	✓	✓	✓	✓	9	Legitimate
NOD32	Legitimate	✓	✓	✓	✓	✓	✓	✓	✓	✓	9	Legitimate
G Data	Legitimate	✓	✓	✓	✓	✓	✓	✓	✓	✓	9	Legitimate
Norton	Legitimate	✓	✓	✓	✓	✓	✓	✓	✓	✓	9	Legitimate
Kaspersky	Legitimate	✓	✓	✓	✓	✓	✓	✓	✓	✓	9	Legitimate
Panda	Legitimate	✓	✓	✓	✓	✓	✓	✓	✓	✓	9	Legitimate
Adware Bot	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Anti Spy Safeguard	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Anti-Spyware	Fake/Crude	✓	X	X	X	✓	✓	X	✓	✓	5	Fake/Crude
Anti Spyware Bot	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Anti-Virus Elite	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Anti Spyware Expert	Fake/Crude	✓	X	X	X	X	✓	X	X	✓	3	Fake/Crude
Error Sweeper	Fake/Crude	X	X	X	✓	X	X	X	X	X	1	Fake/Crude
Major Defense Kit	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Malware Removal Bot	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Netcom3	Fake/Crude	✓	X	✓	✓	X	✓	X	✓	X	5	Fake/Crude
Onescan	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Patchup Plus	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
PC Privacy Cleaner	Fake/Crude	X	✓	✓	X	✓	✓	✓	X	X	5	Fake/Crude
Peak Protection	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Pest Detector	Fake/Crude	X	X	X	X	X	X	✓	✓	✓	3	Fake/Crude
Privacy Control	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Red Cross	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Reg Clean	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Registry Smart	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Security Antivirus	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Protect Code	Fake/Crude	X	X	X	X	X	✓	X	X	✓	2	Fake/Crude
Security Defender	Fake/Crude	✓	X	X	✓	X	✓	✓	✓	✓	6	Fake/Crude
Security Shield	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Security Tool	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
System Security	Fake/Crude	✓	X	✓	✓	X	✓	X	✓	X	5	Fake/Crude
Virus Remover 2008	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
Virus Remover 2009	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
XL Guarder	Fake/Crude	X	X	X	X	X	X	X	X	X	0	Fake/Crude
XP AntiSpyware 2011	Fake/Crude	✓	✓	X	X	✓	X	✓	✓	X	5	Fake/Crude
XP AntiSpyware 2012	Fake/Crude	X	X	X	✓	X	✓	✓	✓	X	4	Fake/Crude
XP AntiVirus 2011	Fake/Crude	X	✓	X	✓	✓	✓	✓	✓	X	6	Fake/Crude
XP AntiVirus 2012	Fake/Crude	X	X	X	X	X	X	✓	✓	X	2	Fake/Crude
XP HomeSecurity 2011	Fake/Crude	X	X	✓	X	X	✓	X	X	X	2	Fake/Crude
XP HomeSecurity 2012	Fake/Crude	X	X	✓	X	X	✓	✓	✓	X	4	Fake/Crude
XP InternetSecurity 2011	Fake/Crude	X	X	X	X	X	X	✓	✓	X	2	Fake/Crude
XP InternetSecurity 2012	Fake/Crude	X	X	✓	X	X	✓	X	X	X	2	Fake/Crude
XP Security 2011	Fake/Crude	X	✓	✓	X	X	✓	✓	X	X	4	Fake/Crude
XP Security 2012	Fake/Crude	X	X	✓	X	X	X	✓	✓	X	3	Fake/Crude
XP TotalSecurity 2011	Fake/Crude	X	X	X	X	X	X	✓	✓	X	2	Fake/Crude

"✓" means there is statistical significance. "X" means there is not statistical significance.

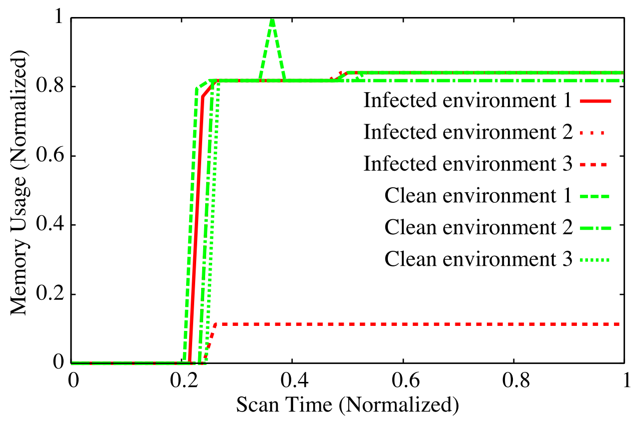


Fig. 10 Memory usages of Pest Detector. The memory usage in infected environment 3 is different from the others. Memory usage and scan time are normalized.

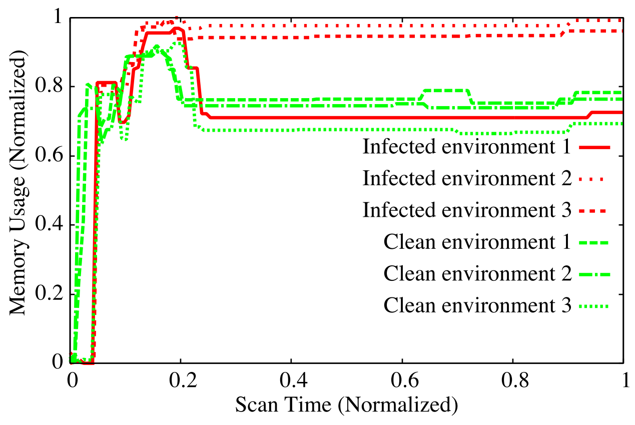


Fig. 11 Memory usage of XP AntiVirus 2011. Each memory usage is different whenever it executes rogue malware scanning. Memory usage and scan time are normalized.

pared with Fig. 9 and Fig. 10, the memory usage patterns show wider divergences. Probably, this fake/crude AV changes its behavior every time it is executed. Since our approach compares the behaviors in clean and infected environments, fake/crude AVs cannot evade our approach even if they change their behaviors randomly. A fake/crude AV must distinguish clean and infected environments to evade our approach.

5. Discussion

5.1 Evasion

It is useless to change memory usage at random to evade our indicator. If memory usage is measured only once in clean and infected environments, the randomly changing memory usage could evade our indicator. However, as explained in Section 3.4, memory usage is measured *M* times in our approach and the Levene Test is performed multiple times. To deceive our approach, fake/crude AV samples must change their memory usage based on the presence of malware. This means that the fake/crude AV would have to act as legitimate ones; that is, they would correctly detect malware infections.

One possible approach to evade our indicator is to use open source AV or leaked source code of legitimate AV. Fake/crude AV samples based on legitimate AV could evade our indicator because their behavior is similar to legitimate AV. However, we believe our indicator raises the bar to developing fake/crude AV because fake/crude AV developers require the source code of product-quality legitimate AV. We also hope vendors can quickly develop effective signatures to detect fake/crude AV based on their products, since the legitimate vendors have the source code of their products and deeply understand the internal behavior of their products.

5.2 Deployment Scenario

Our approach can serve to prevent software download sites from distributing fake/crude AV. Software download sites such as CNET [1] and PCMAG [3] should not distribute fake/crude AV. In spite of their careful management, though, CNET distributed a sample of fake AV in the middle of September 2012 [7]. A discrimination system based on our indicator can prevent the users of those sites from downloading fake/crude AV. Since AV software is usually indexed by tags such as “antivirus” in software download sites, all the pieces of software indexed by “antivirus” can be tested to decide if they are legitimate or fake/crude AV.

Rajab et al. [22] present an analysis of fake AV distribution sites. According to the result, fake AV attacks occur frequently via web sites likely to reach more users including spam we sites an on-line Ads. To detect fake/crude AV distribution sites, there are some approaches such as deSEO [11] and SURF [16]. They detect search poisoning [6] which guides users to malicious web sites. Our approach can identify AV like binaries obtained from deSEO and SURF as fake/crude AV or legitimate AV.

5.3 Other Possible Indicators

We investigated three indicators, file access patterns, CPU usage and memory usage, and found that memory usage is a good indicator to distinguish fake/crude AV from legitimate AV. However, since our indicator needs to collect memory usages multiple times for distinction, the cost may be expensive.

To solve this problem, exploiting file access patterns serves to limit candidates as a pre-filter, because file access patterns of most fake/crude AVs differ from those of legitimate AV as shown in Section 3.3.1. We plan to collect more fake/crude AV samples and discuss whether this method serves to reduce candidates.

6. Related Work

Recently, two studies have reported long-term analyses of fake/crude AV threat ecosystems. They show the traditional signature-based and blacklist-based approaches are useless against fake/crude AV. Rajab et al. show it is practically impossible to keep signatures with a high detection rate against fake/crude AV [22]. The detection rate rises and falls frequently. Cova et al. show that neither IP nor domain-based blacklists are effective on fake/crude AV [5]. Legitimate web sites are often blocked in IP-based blacklists, and domain-based blacklists are evaded by rotating short-lived domains.

Stone-Gross et al. suggest that credit-card companies should endeavor to identify fake/crude AV companies [25]. Fake/crude AV companies monitor the refunds that customers demand from their credit card providers, and they control these refunds so as to keep the chargeback rates low. However, this behavior leads to unusual patterns in chargebacks, which may be leveraged by credit-card companies to identify and ban fraudulent companies.

A white paper has been published to identify fake/crude AV by visual inspection [12]. It provides diverse characteristics about fake/crude AV. By using it, computer users can identify fake/crude AV by visual inspection. As described in Section 2.3, some fake/crude AV samples do not have such characteristics.

Behavioral differences are used to detect various types of malware infection. Egele et al. detect spyware using the dynamic analysis of information flow in Internet Explorers’ plugins [8]. They track information flow of sensitive data and detect spyware if they are leaked to the outside of Internet Explorer. Gu et al. propose BotSniffer which employs several correlation and similarity analyses to identify the crowd of hosts [10]. Gu et al. leverage the fact that bots are likely to conduct malicious activities at the same time.

7. Conclusion

Fake/crude AV software masquerades as a legitimate AV software product with the goal of deceiving victims into purchasing it that seemingly removes malware from their computers. The threat of fake/crude AV is increasing; fake AV accounted for 15% of all malware Google detected on the web [22].

In this paper, we searched for an indicator that captures behavioral differences between legitimate AV and fake/crude AV. The key idea is “fake/crude AV does not consume computer resources when it accesses malware.” Through our experimental investigation, memory usage would be a good indicator because fake/crude AV does not show statistical differences in memory usage between clean and infected environments, but legitimate AV shows clear differences in memory usage. We have demonstrated Levene Test on the memory usages in a clean and an infected environment distinguishes fake/crude AV from legitimate AV. Our method correctly identified 8 out of 8 legitimate AV products and 39 out of 39 fake/crude AV samples.

For future directions, we are planning to extend our approach to be incorporated into anomaly detection systems. Since the scanning behavior of fake/crude AV differs from sample to sample, it is not straightforward to extract a normal profile for the scanning behaviors. Further analysis on the behaviors of legitimate AV would reveal more inherent behaviors that are peculiar to genuine AV, which would enable the anomaly-based detection of fake/crude AV. It would be also interesting to extend our method to on-line or incremental detection of fake/crude AV.

References

[1] CNET | Download.com, available from ⟨http://download.cnet.com⟩ (accessed 2013-07).

[2] FakeAVs (Dashke's blog), available from ⟨http://www.fakeavs.com⟩ (accessed 2013-07).

[3] PCMAG.COM, available from ⟨http://www.pcmag.com/downloads⟩ (accessed 2013-07).

[4] Virus Total, available from ⟨https://www.virustotal.com⟩ (accessed 2013-07).

[5] Cova, M., Leita, C., Thonnard, O., Keromytis, A.D. and Dacier, M.: An Analysis of Rogue AV Campaigns, *Proc. 13th International Symposium on Recent Advances in Intrusion Detection* (*RAID '10*), pp.442–463 (2010).

[6] Doshi, N.: Iframes, Please Make Way for SEO Poisoning (2009), available from ⟨http://www.symantec.com/connect/blogs/iframes-please-make-way-seo-poisoning⟩.

[7] Doyle, S.: How To Remove RegGenie Rogue Antivirus Software – Uninstall RegGenie Malware (Identity Theft Protection) (2012), available from ⟨http://botcrawl.com/how-to-remove-reggenie-rogue-antivirus-software/⟩ (accessed 2013-07).

[8] Egele, M., Kruegel, C., Kirda, E., Yin, H. and Song, D.: Dynamic Spyware Analysis, *Proc. USENIX Annual Technical Conference*, pp.233–246 (2007).

[9] Fossi, M., Turner, D., Johnson, E., Mack, T., Adams, T., Blackbird, J., Low, M. K., McKinney, D., Dacier, M., Keromytis, A.D., Leita, C., Cova, M., Orbeton, J. and Thonnard, O.: Symantec Report on Rogue Security Software (2009), available from ⟨http://www4.symantec.com/Vrt/wl?tu_id=TeCm125590003756772344⟩ (accessed 2013-07).

[10] Gu, G., Zhang, J. and Lee, W.: BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic, *Proc. 15th Annual Network and Distributed System Security Symposium* (*NDSS '08*) (2008).

[11] John, J.P., Yu, F., Xie, Y., Krishnamurthy, A. and Abadi, M.: deSEO: Combating Search-Result Poisoning, *Proc. 20th USENIX Security Symposium*, pp.299–313 (2011).

[12] Karnik, A., Rico, A.C.J., Prakash, A. and Honjo, S.: Identifying Fake Security Products (2009), available from ⟨http://www.mcafee.com/us/resources/white-papers/wp-identifying-fake-security-products.pdf⟩ (accessed 2013-07).

[13] Kiguolis, U.: Remove Security Antivirus (2010), available from ⟨http://www.2-spyware.com/remove-security-antivirus.html⟩ (accessed 2013-07).

[14] Kiguolis, U.: Remove Anti-Virus Elite (2012), available from ⟨http://www.2-spyware.com/remove-antivirus-elite.html⟩ (accessed 2013-07).

[15] Levene, H.: *Robust Tests for Equality of Variances*, Stanford University Press (1960).

[16] Lu, L., Perdisci, R. and Lee, W.: SURF: Detecting and Measuring Search Poisoning, *Proc. 18th ACM Conference on Computer and Communications Security* (*CCS '11*), pp.467–476 (2011).

[17] MDL: Malware Domain List, available from ⟨http://www.malwaredomainlist.com/⟩ (accessed 2013-07).

[18] Oberheide, J., Cooke, E. and Jahanian, F.: Cloud AV: N-Version Antivirus in the Network Cloud, *Proc. 17th USENIX Security Symposium*, pp.91–106 (2008).

[19] Open Malware: Open Malware — Community Malicious code researh and analysis, available from ⟨http://offensivecomputing.net⟩ (accessed 2013-07).

[20] Paget, F.: Running Scared: Fake Security Software Rakes in Money Around the World (2010), available from ⟨http://www.mcafee.com/us/resources/white-papers/wp-running-scared-fake-security-software.pdf⟩ (accessed 2013-07).

[21] Provos, N., Mavrommatis, P., Rajab, M.A. and Monrose, F.: All Your iFRAMEs Point to Us, *Proc. 17th USENIX Security Symposium*, pp.1–16 (2008).

[22] Rajab, M.A., Ballard, L., Mavrommatis, P., Provos, N. and Zhao, X.: The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution, *Proc. 3rd USENIX Workshop on Large-Scale Exploits and Emergent Threats* (*LEET '10*) (2010).

[23] Sophos: What is FakeAV? (2010), available from ⟨http://www.bt.bt/forms/sophos-what-is-fakeav-wpna.pdf⟩ (accessed 2013-07).

[24] Sophos: Stopping Fake Antivirus: How to Keep Scareware off Your Network (2013), available from ⟨http://www.sophos.com/en-us/medialibrary/Gateddf⟩ (accessed 2013-07).

[25] Stone-Gross, B., Abman, R., Kemmerer, R.A., Kruegel, C., Steigerwald, D.G. and Vigna, G.: The Underground Economy of Fake Antivirus Software, *Proc. 10th Workshop on Economics of Information Security* (*WEIS '11*) (2011), (online) available from ⟨http://weis2011.econinfosec.org/papers/The Underground Economy of Fake Antivirus Software.pdf⟩.

[26] Trend Labs: Unmasking FAKEAV (2010), available from ⟨http://solutionfile.trendmicro.com/solutionfile/EN-1055340/EN/Unmasking_FakeAV_v4.pdf⟩ (accessed 2013-07).

[27] Wu, Z., Gianvecchio, S., Xie, M. and Wang, H.: Mimimorphism: A New Approach to Binary Code Obfuscation, *Proc. 17th ACM Conference on Computer and Communications Security* (*CCS '10*), pp.536–546 (2010).

Masaki Kasuya received his B.E. and M.E. degrees from Keio University in 2009 and 2011, respectively. He is currently a Ph.D. candidate in Keio University. His research interests include malware security, browser security, and system security. He is a student member of IEEE, ACM and IPSJ.

Kenji Kono received his B.Sc. degree in 1993, M.Sc. degree in 1995, and Ph.D. degree in 2000, all in computer science from the University of Tokyo. He is an associate professor of the Department of Information and Computer Science at Keio University. His research interests include operating systems, system software, and Internet security. He is a member of IEEE/CS, ACM and USENIX.