ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

SeCom: A Novel Approach for Malware Confiscation in OS level Virtual Machines

R. Hema[1], T. VetriSelvi[2]

Final Year, M.E. Computer Science and Engineering, K. Ramakrishnan College of Technology, Trichy, , Tamilnadu, India[1]

Asst. Prof (Head of the Department of CSE), K. Ramakrishnan College of Technology, Trichy, Tamilnadu, India[2]

ABSTRACT: Virtual Machines are based on the specifications of a presumptive computer. It is an independent instance and performs the function as like the original host machine. It can be created upon use and disposed upon the completion of the tasks or the detection of error. One of the main demerits of virtual machine is that if there is no malicious activity, the user has to redo all of the work in her actual workspace since there is no easy way to commit. So, a lightweight commitment approach called SeCom have been proposed, which eliminates the malicious program at the end of virtual machine termination i.e. while committing the benign data. It consists of three steps: correlation, recognition and commitment. Firstly, instead of manipulating huge data, it relies only on the OS level information flow and malware behaviors, thereby it reduces performance overhead. Secondly, it recognizes the data in cluster by cluster manner, to ease the detection. Thirdly, it marks the cluster as harmful if and only if it has at least two different types of malware behavior, to reduce the false positives. When comparing with other commercial antimalware tools, it cleans up all the malware behavior and maintains the performance of host machine to the desired level. Moreover, it results in lower number of false alarms than that accomplished by behavior based approach of antimalware tools.

KEYWORDS: Virtual Machine, malware deeds, false positive, false negative, benign process, secure commitment, raw behavior pair.

I. INTRODUCTION

An OS level virtual machine is a cost- effective component due to its minimal startup/shutdown cost. Also it requires less resource and supports high scalability due to its sharing of the execution environment of the host operating system and confining state changes within the VM's environment. It is thus excellent equipment for tolerating intrusions and faults, as well as consolidating servers. A practical application is to allow users to install and try new applications without worrying about malware. In other words, if something abnormal happens, one can simply throw away the infected VM. But the problem of virtual machine is there is no malicious activity detection only the benign updates send to the virtual machine host environment i.e. the user has to repeat all the activities in her actual user space since there is no commitment program. Changes within an OS-level VM include files, directories and registry entries that are created, modified and deleted by the processes running in the VM. Secure commitment means merging only benign changes into the host environment but filtering out malicious changes when committing a VM.

There are two issues to address in order to build a secure commitment mechanism in the framework of an OS-level virtual machine. First, the overhead of a commitment mechanism imposed on the host OS should be as low as possible since the virtual machine mechanism has already incurred no trivial overhead which leads to the performance degradation. Second, a commitment mechanism should be able to clean up all malicious changes rather than part of them. However, existing technologies such as logging and analysis, host-based intrusion detection and anti-malware can not address the two issues simultaneously. These techniques either cannot identify all malicious changes made by a malware program or incur a big overhead on a system although they may be effective in detecting intrusions. Therefore, a light-weight commitment approach, named SeCom has been proposed for an OS-level virtual machine to prevent malicious changes from being merged into the host. To our best knowledge, this is the first effort towards building a secure and practical VM commitment mechanism. It automatically filters out malware impacts when committing the content of a VM into the host environment. Thus, a user does not need to manually scan the VM to be committed using antivirus software each time. Moreover, a user also does not need to install, manage and frequently upgrade the anti-virus software. Therefore, this approach is useful when building self-healing or self-protection systems based on VMs.

The approach consists of three stages. First, it correlates suspicious objects within a VM into a number of clusters by tracking OS-level information flows and attaching a cluster label to each object. Objects in a cluster are only possible to be either all benign or all malicious. Second, it determines malicious clusters using an online malware detection engine to monitor malicious behaviors. Last, it merges benign changes in a VM to the host environment while discarding malicious clusters. It has three novel features. First, unlike the commitment approaches assumed in other fields (e.g. database) which rely on a huge volume of log data, it leverages use of OS-level information flows and malware behaviors to perform secure commitment. As a result, SeCom imposes a smaller overhead on host OS, while using a conventional data-logging method would significantly slow down the whole system. Second, different from existing intrusion detection and recovery systems that detect compromised OS objects one by one, it puts correlated objects into clusters thus identifying and discarding compromised objects cluster by cluster. Finally, different from existing behavior-based malware detection methods, it monitors a pair of malware behaviors and labels the sources of the processes that launch the behaviors rather than monitors a single behavior. SeCom approach depends only on tracing OS-level information flows and monitoring malware behaviors without the need of technical details of a specific virtual machine. Therefore, although SeCom is designed for OS-level virtual machines, with some changes it should also be applicable to other types of virtual machines or general operating systems in order to clean up malware impacts.

II. OVERVIEW OF SECOM APPROACH

Committing a VM overwrites files and registries on the host with the VM's private versions. As malware contributes to most security problems, to protect the integrity of the host environment, files and registries that have been attacked by malware programs should be discarded when committing the VM. The design of SeCom is based on results obtained from our preliminary study o f malware behaviors. To find an approach to identify malware objects from the contents of a VM, we have analyzed the technical details of a large number of malware samples from Symantec Threat Explorer [2] that stores analysis results of thousands of malware samples by analysts. With the study results, a novel approach, SeCom has been designed and developed, to commit VM. It mainly leverages light-weight techniques such as tracing OS-level information flows and monitoring malware behaviors to ensure secure commitment, rather than uses logging technique which often incurs significant storage and time overhead, and even requires a backend host. SeCom consists of three steps, i.e. "correlate", "recognize" and "commit", which can be conceptually depicted in Fig. 1. The first step correlates suspicious OS objects within a VM that are potentially malicious into different clusters. The second step recognizes real malicious clusters and marks them. The third step commits all OS objects in a VM to the host except the ones in malicious clusters or changes made on write-protected files.

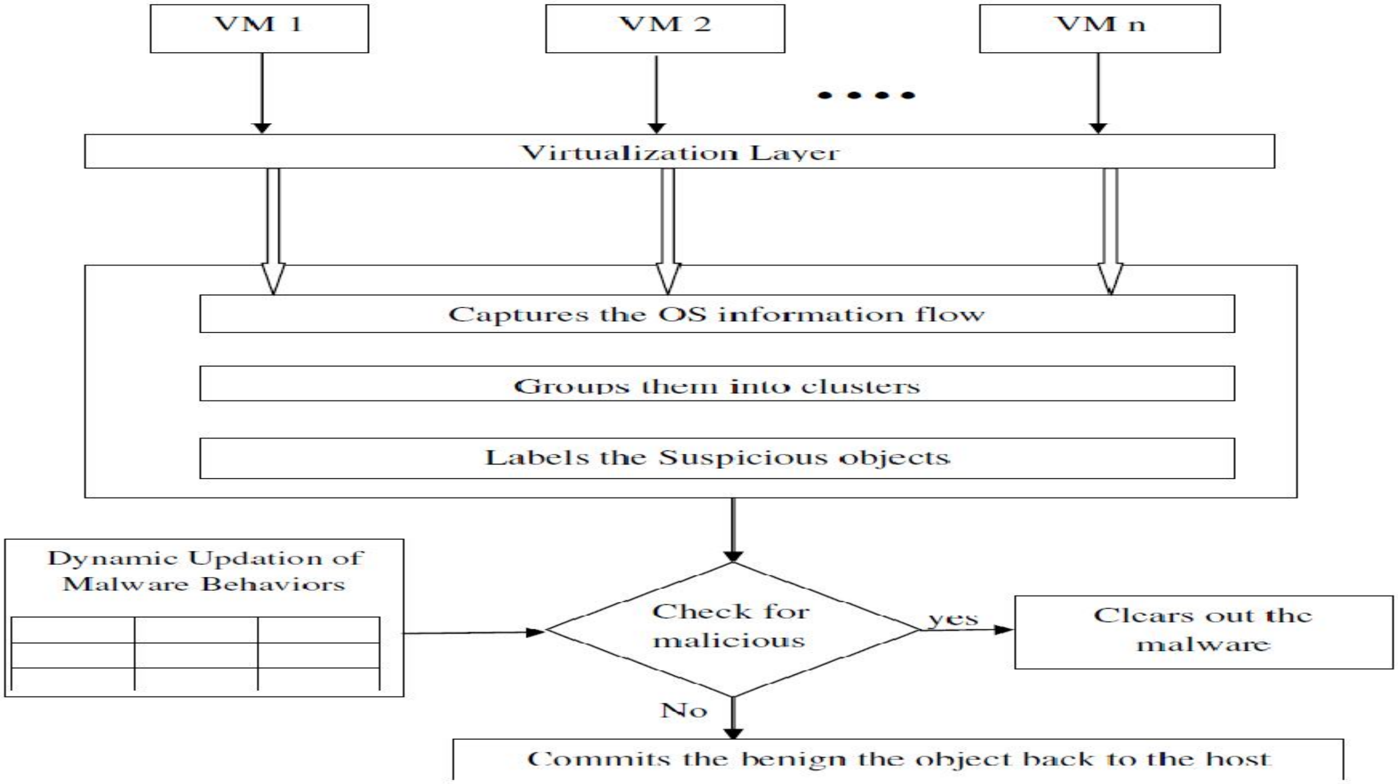

VM 1
VM 2
VM n
Virtualization Layer
Captures the OS information flow
Groups them into clusters
Labels the Suspicious objects
Dynamic Updation of Malware Behaviors
Check for malicious
yes
Clears out the malware
No
Commits the benign the object back to the host

Fig. 1. SeCom Approach

A VM can only be committed when it has completed all the tasks and is at the stage of being shut down, because many objects and processes within a running VM cannot be merged into the host environment. For example, some objects (e.g., files) are often locked when accessed by some processes. In addition, the running of most processes often depends on some kernel objects, interprocess communication objects or process properties that are tied with a specific VM. Moreover, committing a running VM may result in a partial merge of results from a task still being performed. Therefore, the "correlate" and "recognize" steps are executed when a VM is running, while the "commit" step is only executed after a VM is stopped. In the rest of this section, we describe the three steps involved in securely committing a VM.

A. *Correlating Suspicious Objects*

One novel feature of our VM commitment approach is to identify malicious OS objects in a cluster fashion for a more efficient commitment, rather than one by one as done in traditional malware detection and analysis methods. Moreover, it is also able to remove malicious objects more completely, because malware programs generate or modify a non-trivial number of files or registry entries on a single OS and only removing part of the malware program thoroughly. To identify malicious objects in a cluster fashion, first of all, we have to address the challenge of correlating suspicious objects into clusters. Since objects of a malware program often have various types and are scattered all over the system, it is difficult to associate them together. We observe that objects of a malware program can be correlated together by tracing information flows, and at the same time the malicious objects can be clearly separated from the other objects through a proper way of attaching cluster labels to them. Accordingly, we devise a novel approach to correlate suspicious objects into clusters, which includes tracing and labeling suspicious objects.

1) *Tracing Suspicious Objects:*

As all malware programs come from either the network or removable drives, we treat the following objects as start-points to trace suspicious objects, calling them start-point objects.

- Processes conducting remote communication
- Executables (i.e. executable file) located at removable drives.

An executable represents an executable file with a specific extension, such as .EXE, .COM, .DLL, .SYS, .VBS, .JS, BAT, etc, or a special type of data file that can contain macro codes, say a semi-executable, such as .DOC, .PPT, .XLS, .DOT, etc. SeCom does not allow a suspicious process to change the extension of a file in order to prevent its potential evasion of tracing. With these two rules, all malware programs that attempt to enter the system can be tracked as there are only two ways for them to break into system, either through network communications or through a removable drive. To track OS level information flow, BackTracker [1], is a successful approach. However, the major challenge is how to make sure that the entire system does not get marked as suspicious and at the same time malware programs cannot evade being traced. This actually requires a trade-off between reducing the number of marked objects and reducing the risk of malware evasion. Our principle to achieve the trade-off is to trace preferentially the information flows with a high risk of propagating malware programs while not tracing the information flows with a low risk. Based on this principle, we mark the following objects as suspicious.

- Files, directories and registry entries created or modified by a suspicious process
- Processes spawned by a suspicious process;
- Processes loading a suspicious executable file or reading a suspicious semi-executable or script file.

The first rule records all permanent changes in a VM made by suspicious processes so that maliciously changed application data, executable files, system configurations, directories, registry entries and so on can be filtered out thoroughly when committing a VM. To track the information flows with a high risk of propagating malware programs, the last two rules focus on tracing executables and processes. As an executable represents an inactive malware while a process represents an active malware, the information flows presented in these three rules have a high possibility of propagating malware programs. In the third rule, semi- executable and script file possibly contain malware programs (e.g., macro virus in MS Word), and thus the processes reading them need to be marked. Although the macro virus protection in Office software can reduce the chances of macro virus infection, relying on it is very dangerous as crafted macro codes are able to subvert it and cause destructive damages.

2) *Labeling Suspicious Objects*

In this section, the dependency graph has been employed to describe how to attach cluster labels to suspicious objects. Actually, for each start-point object, its descendent objects are connected to each other by information flows and form an existent but invisible dependency graph, which had been disclosed by the literature work [1]. The graph is a directed graph and has the start-point object as its root node. Its nodes represent OS objects, e.g. file, process. Its edges represent information flow related operations, e.g. creating a process, modifying a file. Figure 2 (a) and (c) show two dependency graphs which are derived from a networking process and an executable file respectively. Note that, we do not intend to really generate dependency graphs to help label objects since this would not be applicable to an online approach. Instead, the labeling methods are implemented together with the starting and tracing rules as follows: when an object is determined as suspicious by starting or tracing rules, a proper cluster label, i.e. a number and a time stamp, will be attached to it at the same time in order to denote that it is a suspicious object and belongs to the cluster identified by the label. In other words, the labeling methods are enforced along with the starting and tracing rules in real-time, rather than generating a dependency graph and then analyzing it. When a start-point object is a network facing process, its dependency graph is too coarse grained to be used to recognize malicious objects in a cluster fashion since it might contain both benign and malicious objects. In other words, we cannot determine that all objects in a graph are malicious even if most of the objects in the graph are recognized as malicious. Thus, the graph is partitioned into a number of sub graphs, say clusters, so that each cluster contains either only benign or only malicious objects.

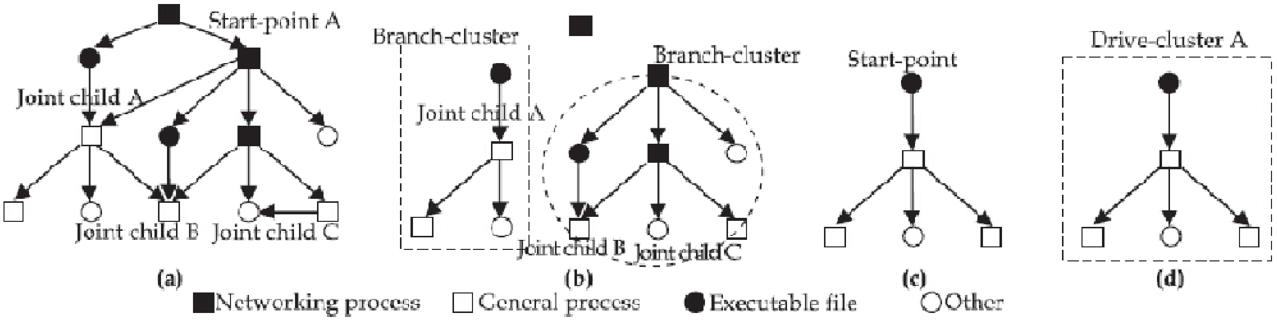


Fig. 2. Dependency graph samples and obtained clusters

The malware programs break into a host through three basic attack channels. The first is that, malware programs exploit bugs in network-facing daemon programs or client programs and compromise them, then immediately spawn a shell or back-door process [3]. After this, the attacker typically tries to download and install attacking tools and rootkits, as well as performs any other adversary actions. Accordingly, we give a cluster label to a process directly spawned by a network- facing process as well as its descendants, calling them a branch cluster, e.g., the branch cluster B in Figure 2 (b). A branch cluster corresponds to a sub graph of a dependency graph which roots from a network facing process. The other attack channel is that, malware programs lure users into downloading and launching them [4]. After started, malware programs copy themselves and make themselves resident in a host. Consequently, we give a cluster label to the downloaded executable and its descendants, calling them a branch cluster as well, e.g., the Branch-cluster A in Figure 2 (b). The last channel is removable drives. Therefore, we give a cluster label to an executable file located in a removable drive and all its descendent objects, calling the formed cluster a drive cluster, e.g., the Drivecluster A in Figure 2(d). Splitting a dependency graph into different branch clusters might cause a piece of malware to be split into two separate processes on different branch clusters, which could work together to perform malicious actions and potentially evade SeCom's detection. We can prevent this evasion at the time to commit a VM.

B. *Recognizing Malicious Clusters*

To recognize a malicious cluster, an on-line engine has been built to monitor whether the processes in the cluster exhibit any malware behaviors. Recent research efforts [5][6] on behavior based malware detection often employ dynamic data flow tracing techniques to extract featured malware behaviors. The racing of dynamic data flow involves a big overhead, which significantly slows down the system and is thus not applicable for on-line monitoring [7]. On the other side, to complement with the traditional signature-based detection, commercial antivirus software often has an online behavior-based malware detection engine. However, the engine identifies a malware program only based on a single suspicious behavior which might also appear in benign software, and thus frequently produces false alarms that distract users [8][5]. Different from existing efforts, our malware detection engine detects malware by combining the techniques of tracing OS-level information flow and online malware detection. More specifically, based on the clusters formulated as a result of tracing OS-level information flow, the engine monitors all the behaviors of processes in a cluster and determines whether a cluster is malicious. A cluster is considered malicious if it exhibits two behaviors that match a predefined raw-behavior-pair, as shown in Fig. 3. A raw-behavior-pair (RBP) consists of two independent raw behaviors. A raw behavior is extracted by intercepting a single system/API call and its parameters. It can differentiate malware from benign programs but may result in a few false positives. For example, "modifying registry for automatic startup" is such a malware behavior.

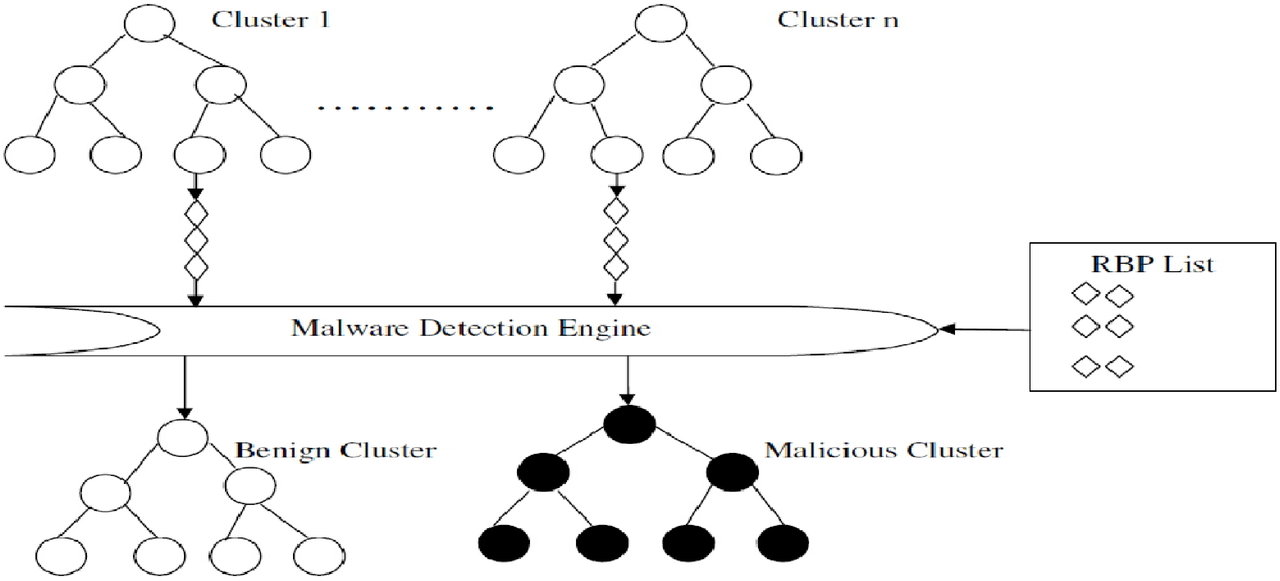


Fig. 3. Malware detection engine

While targeting for online detection, false positive rate can be reduced, when this detection engine uses a RBP to detect malware. This will make the false positive rate much lower than that generated with commercial online detection techniques in anti-virus software [8] which only rely on a single raw behavior to identify malware. If the raw behaviors in a RBP are carefully selected, the FPR of the RBP-based detection will be as low as that of the detection based on the featured behaviors. Moreover, since all clusters derive from dangerous sources, i.e. the network and removable drives, our detection approach actually considers not only malware behaviors but also the sources of the process launching the behaviors when determining a malicious cluster. However, there is a big challenge to realize the system using RBP to detect malware. Commercial online engines [8] cannot achieve the goal because they cannot correlate two behaviors which may exhibit at different time and associate them with a single malware program. For example, two behaviors may be launched by a malware program's two distinct processes respectively. Although data flow tracing techniques [5] can find potential dependency between the behaviors, these techniques will levy unacceptable heavy overhead on the system. Instead, with an intelligent tracing of OS-level information flow, our detection engine can correlate the two processes and then naturally associate the two behaviors together. We determine a cluster as malicious if it exhibits two different types of raw behaviors. After breaking into a system, a malware program generally fulfills malicious tasks through four basic steps:

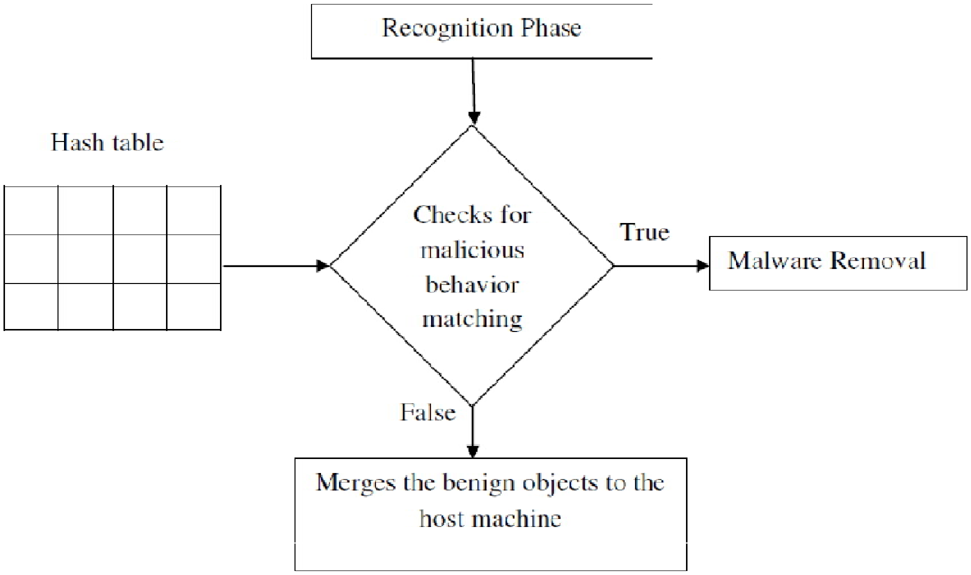
- Making itself to auto-start after system booting,
- Propagating itself across the system,
- Hiding itself from users and antimalware tools,
- Achieving malicious goals.

Each step can be fulfilled by a type of behaviors. Accordingly, malware behaviors can be divided into four types. Although not every malware program requires all the four types of behaviors, a malware program does invoke at least a few types of the behaviors. Therefore, a RBP can be constructed with any two raw behaviors that belong to different types respectively. This way, we do not need to maintain a long RBP list to search, but only make detection based on the types of the behaviors. A crafted malware program might exhibit only one type of raw behaviors instead of two to avoid matching any RBP. Then, it waits until it is committed to the host environment to perform malicious behaviors. Only by setting up an ASEP, a malware program can make itself resident in a host. To fight against such malware programs, at the committing stage, we will discard any changes on ASEP if the corresponding cluster is not derived from a trusted source. Thus, such malware programs will be disabled after the commitment without setting an ASEP in advance. If the process launching the current raw behavior does not belong to any cluster, i.e., the process is not derived from dangerous sources including the network and removable drives, the engine will not trigger any action. (2) If the process belongs to a cluster and the cluster does not exhibit any raw behavior so far, the engine will record the type of the current behavior into the cluster. (3) If the types of the current and the recorded behaviors of the cluster are different, the engine will mark the cluster as malicious. (4) If the types of the behaviors are the same, the engine will not take any action. (5) If the current behavior is a bypassing behavior, the engine will mark the cluster as malicious and at the same time refuse it.

C. Committing the Benign Cluster

When a VM is stopped and the user requires deleting or committing the VM, SeCom invokes the commitment function. Since the VM has been terminated, there is not any pending task or job in the VM. Meanwhile, all of the processes and other volatile objects, e.g., IPC objects, within a VM have already been erased from the OS and thus do not need to be committed. Only the permanent objects, e.g. files, directories, registry entries, need to be considered. The commitment procedure is completed following three steps. First, all benign clusters are checked and a benign cluster has been marked as malicious if it contains an ASEP but does not derive from a trusted source. Thus, we can prevent the potential evasion to SeCom mentioned above, i.e., a malware program merely sets up an ASEP and waits for the commitment to execute the rest of the behaviors in the host environment. We introduce a Remote Administration Point (RAP) to represent a trusted source. A RAP is a special application dedicated to install software or manage the system from the remote. It applies the principle of diversity [12][13] and integrity protection techniques. Specifically, we install two different forms of programs with the same function, e.g., different kinds of web browsers. One is for daily use while the other is for RAP. Thus, we can set tight restrictions on the RAP program without affecting usability since one can use the other program. The RAP program is configured to have the highest security protection level, and only communicate with a few remote hosts through secure protocols. Moreover, SeCom discards any changes made on the configurations and files of the RAP program so as to strictly preserve the integrity of the RAP program.

Second, every ASEP in malicious clusters has been checked to see whether the corresponding auto-start executables are placed in other benign clusters, and then mark such benign clusters as malicious. Thus, we can completely remove the malware programs that intentionally distribute ASEP hooks and the auto-start executables into different clusters. Third, all the objects in malicious clusters are discarded and merge the objects in benign clusters into the host. Meanwhile, the objects not included in any cluster also need to be merged as they are not derived from the network or removable drives and thus benign. Fig. 5 shows the commitment of benign updates to host machine.


Recognition Phase
Hash table
Checks for malicious behavior matching
True
Malware Removal
False
Merges the benign objects to the host machine

III. EVALUATION

To demonstrate the applicability of the OS-level VM commitment approach, we have successfully developed a prototype under the framework of Feather- Weight Virtual Machine [14], which partitions the name space of a single Windows OS to form a number of virtual machines.

A. Secure Commitment

To recognize malicious clusters, we configured following raw behaviors into the detection engine: Type (I): Modifying registry for auto-startup, Creating or modifying Windows services, Installing or modifying Windows drivers. Type (II): Self replication, Injecting into other processes, Creating processes abnormally. Type (III): Modifying registry to hide its presence, lowering security settings, disabling the host Firewall, Killing anti-malware processes, Compromising antimalware files or settings, closing system restoring mechanism. Type (IV): Stealing confidential information. For every malware sample, we perform a three-step experiment. First we run the malware sample in a newly created VM without turning on the SeCom and record what objects it creates or modifies. Then, we enable the SeCom, run the same malware sample and other arbitrary benign applications together in a new VM, and eventually commit the VM. When performing the commitment, we make the VM committing module to print out the names of discarded objects. Lastly, we run the benign applications committed in the host environment in order to check whether the commitment process damages the internal consistency of the benign applications. The false positives are resulted because two benign programs exhibited behaviors of type(I) and (IV), i.e., modifying registry for auto-startup and reading sensitive files. Hence, the FN rate and FP rate of SeCom are 0% and 4% respectively. Moreover, all of the benign applications which were mixed together with the malware samples can work smoothly after being committed into the host environment. In other words, the commitment process does not have impact on the internal consistency of the benign changes that coexist with malicious ones in the same VM.

B. Comparison with Commercial Tools

To further evaluate SeCom, another experiment has been performed using two popular commercial anti-malware tools: Kaspersky and VIPRE. First, we use both commercial tools to scan the commitment results of SeCom in the host environment after running a malware sample. For each sample, neither commercial tool could detect the malware in the host environment. Thus, we conclude that SeCom has removed the malware to the satisfaction of the commercial tools. Second, we test all of the samples, using the signature-based module and behavior-based module of every antimalware tool. Fig. 5 shows the FP rates obtained from running five categories of benign samples. S and B represent signature based module and behavior-based module respectively. From the Fig. 6, FP rate of SeCom is slightly higher than that of signature based modules while much lower than that of behavior-based modules. The reason is that, behavior-based technique often causes a higher FP rate than signature-based scheme (which cannot detect unknown malware programs), but considering double behaviors and the originators of the processes exhibiting the behaviors will dramatically reduce FPs. Fig 6 shows the FN rates of detecting five categories of malicious samples. In the figure, both commercial tools only detect 50~75% of malicious changes regardless of what techniques they use. However, SeCom identifies all of the changes imposed by different categories of malicious samples. In other words, SeCom successfully clears all malicious changes when committing a VM.

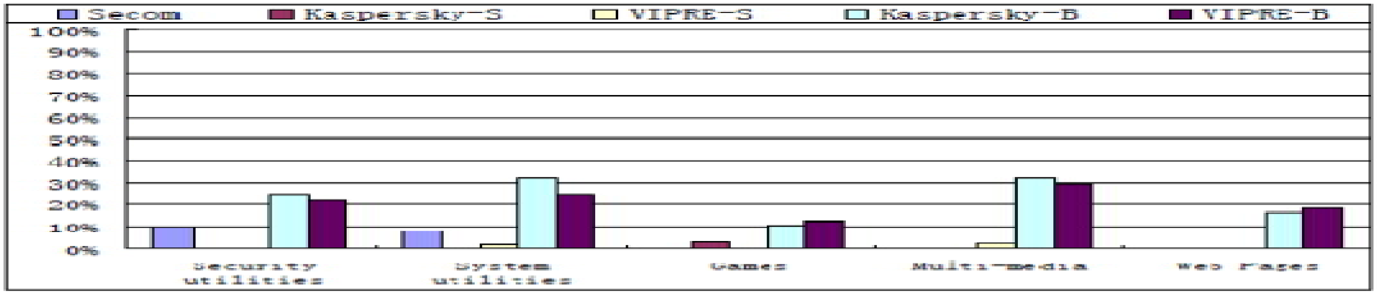


Fig 6. Detecting results of the benign samples using commercial tools and SeCom

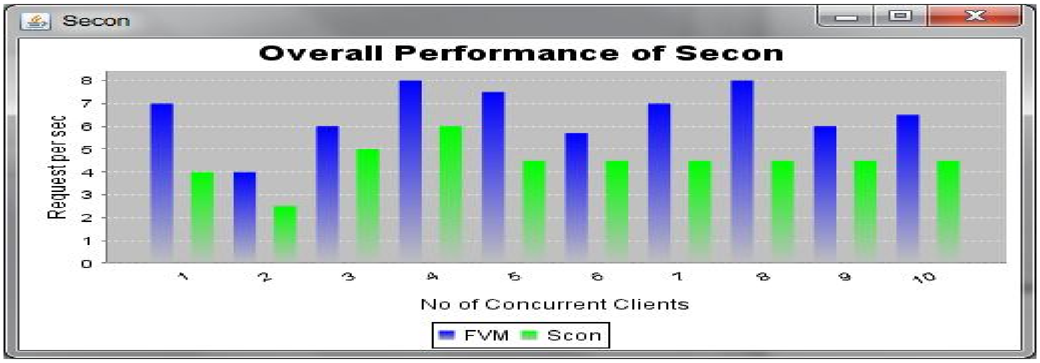


Fig. 7 Detecting results of the changes made by malware samples using commercial tools and SeCom

IV. RELATED WORK

There is no such a project that can securely commit VM on an OS level virtual machine platform in the literature. Popular OS-level VM technologies, e.g. FreeBSD Jail [15], Linux-VServer [16], Solaris Zones [17], OpenVZ [18] and Virtuozzo [19], do not provide the functionality to securely commit VM. Feather- weight Virtual Machine (FVM) [14] only checks and abandons the resource updates under the security related file directory and registry entry within a VM to eliminate side effects left by malicious programs before VM commitment. However, simply examining special resource updates confined inside a VM is not sufficient to detect all the suspicious behaviors or to recognize the exact scope of the attack due to malware execution. Sun et al. [20] present an approach for realizing a safe execution environment (SEE) that enables users to "try out" new software without the fear of damaging the system in any manner. It implemented a commitment approach to ensure semantic consistency of the committed results. Before commitment, user needs to make his own decision on whether the running results contained in a SEE are safe to commit. SeCom, however, aims at automatically identifying unsafe results inside a VM before commitment. Hence, SeCom differs from paper [20] and can be a complementary technology to [20] to help recognize unsafe results.

As an alternative technology, system call log analysis is able to detect compromised system resources and prevent them from being committed. Research efforts demonstrate the potential capability of log analysis to securely commit VM since log analysis can identify compromised files or derive malicious process behaviors [21][22][23]. However, the enforcement of log recording and analysis often significantly slows down the system which makes it very inefficient and possibly unusable. Former work [24] also traces OS-level information flow, but it aims to block critical malware behaviors instead of committing safe changes in a VM back to the host. It also does not further correlate objects into clusters which will facilitate the fast commitment. SeCom is also different from other candidate technologies, such as host-based intrusion detection and antimalware, though all of them are able to recognize malicious objects. SeCom aims to recognize all side effects of a malware program imposed on a system while intrusion detection and anti-malware technologies typically are interested in determining whether a single file or process is adverse. A recent approach proposed in [24] can remove all effects of a malware program, but requires a training stage to generate a remediation program for cleaning the impacts a specific malware. Therefore, SeCom meets the requirement of secure commitment better than these candidate technologies.

V. CONCLUSION

SeCom, a scheme towards securely committing OS level virtual machines has been proposed, which is required by intrusion-tolerant applications and system administrations to save benign changes within a VM to the host environment. So far, none of the publicly available documents on OS-level virtualization technologies ever provides a feasible scheme to securely commit VM. We thus believe that SeCom is the first secure commitment scheme. The critical challenge behind securely committing VM is to identify compromised objects thoroughly and lightly. To address the challenge, SeCom has been proposed that consists of three steps. First, it correlates suspicious OS objects within a VM together by tracking OS-level information flows and grouping them into clusters by intelligently attaching different labels to objects. Then, it recognizes a malicious cluster by a behavior based malware detection engine. Last, it commits VM while discarding malicious clusters. SeCom has three novel features. First, SeCom can lightly commit VM using OS-level information flows and malware behaviors. Second, SeCom detects and discards malicious changes in a cluster fashion to clean up malware impacts quickly and thoroughly. Finally, to reduce the false positive rate, SeCom considers two malware behaviors which are of different types and the originator of the processes which exhibit the behaviors when identifying a malicious cluster. We implemented SeCom under the framework of FVM and conducted experiments concerning the performance of commitment and overhead. The experiment results demonstrate that SeCom can effectively clean up malware impacts when performing commitment and only causes 0.8% to 8.3% additional performance overhead on system. Moreover, compared with commercial anti-malware tools, it can erase malware more thoroughly and produce a lower false positive rate. Hence, it fits the task of VM commitment better.

REFERENCES

[1] S.T. King and P.M. Chen. Backtracking Intrusions. Proceedings of ACM Symposium on Operating Systems Principles, pages 223–236, October 2003.

[2] Symantec, Inc, http://www.symantec.com/business/security_response/threatexplorer/aaathreats.jsp.

[3] N. Li, Z. Mao, H. Chen, "Usable Mandatory Integrity Protection for operating Systems," IEEE S&P, pages 164– 178, 2007.

[4] M. Egele , P. Wurzinger , C. Kruegel , E. Kirda, Defending Browsers against Drive-by Downloads: Mitigating Heap- Spraying Code Injection Attacks,Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, July 2009, Como, Italy.

[5] H. Yin , D. Song , M. Egele , C. Kruegel , E. Kirda. Panorama: capturing system-wide information flow for malware detection and analysis. CCS2007, pages 116–127, New York, NY, USA, 2007.

[6] E. Kirda, C. Kruegel, G. Banks, G. Vigna, and R. Kemmerer. Behavior based spyware detection. In Usenix Security Symposium, 2006.

[7] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang. Effective and efficient malware detection at the end host. In USENIX Security Symposium, Montr´eal, Canada, August 2009.

[8] O. Sukwong, H. Kim, J. Hoe, "An Empirical Study of Commercial Antivirus Software Effectiveness", Computer, Jun. 2010. IEEE Computer Society.

[9] Y.-M. Wang, R. Roussev, C. Verbowski, A. Johnson, M.-W. Wu, Y. Huang, and S.-Y. Kuo. Gatekeeper: Monitoring autostart extensibility points (aseps) for spyware management. In Proceedings of 18th Large Installation System Administration Conference, November 2004.

[10] Y. Yu, H. K. Govindarajan, L. Lam, T. Chiueh. Applications of Feather Weight Virtual Machine. Proceedings of the 2008 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, Seattle WA, March 2008.

[11] C. Verbowski, E. Kiciman, A. Kumar, B. Daniels, S. Lu, J. Lee, Y.-M. Wang, and R. Roussev. Flight data recorder: monitoring persistent-state interactions to improve systems management. In Proceedings of the 7th symposium on Operating systems design and implementation, pages 117–130, 2006.

[12] B. Littlewood and L. Strigini. Redundancy and diversity in security. In Proceedings of the 9th European Symposium on Research in Computer Security (2004), pages 423–438.

[13] E. Totel, F. Majorczyk, and L. Me. COTS diversity baded intrusion detection and application to web servers. In Eighth International Symposium on Recent Advances in Intrusion Detection, Seattle, Washington, September 2005.

[14] Y. Yu, F. Guo, S. Nanda, L. Lam, T. Chiueh. A Featherweight Virtual Machine for Windows Applications. In Proceedings of the 2nd ACM/USENIX Conference on Virtual Execution nvironments. Pages 24– 34, Ottawa, 2006

[15] P.-H. Kamp and R. N. M. Watson. Jails: Confining the omnipotent root. In Proceedings of the 2nd International SANE Conference, 2000.

[16] S. Soltesz , H. Pötzl , M. E. Fiuczynski , A. Bavier , L. Peterson. Container based operating system virtualization: a scalable, high-performance alternative to hypervisors. In proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, Lisbon.

[17] D. Price and A. Tucker. Solaris Zones: Operating system support for consolidating commercial workloads. In Proceedings of the 18th Large Installation System Administration Conference (LISA 2004), pages 241–

[18] OpenVZ. Unique features of OpenVZ. http://openvz.org/documentation/tech/features.

[19] SWsoft. Virtuozzo server virtualization. http://www.swsoft.com/en/products/virtuozzo.

[20] W. Sun, Z. Liang, V. Venkatakrishnan, and R. Sekar. Oneway isolation: An effective approach for realizing safe execution environments. In Proceedings of 12th Annual Network and Distributed System Security Symposium, 2005, pages 265–278.

[21] F. Hsu, H. Chen, T. Ristenpart, J. Li, and Z. Su. Back to the Future: A Framework for Automatic Malware Removal. In Proc. Annual Computer Security Applications Conference, 2006.

[22] N. Zhu, and T. Chiueh. Design, implementation, and evaluation of repairable file service. In Proceedings of the 2003 International Conference on Dependable Systems and Networks, pages 217–226, June 2003.

[23] O. Sukwong, H. Kim, J. Hoe, "An Empirical Study of Commercial Antivirus Software Effectiveness", Computer, Jun. 2010. IEEE Computer Society.

[24] R. Paleari, L. Martignoni, E. Passerini, D. Davidson, M. Fredrickson, J. Giffin, and S. Jha, "Automatic generation of remediation procedures for malware", USENIX Security Symposium, Washington DC, August 2010.