# Fingerprinting Large Data Sets through Memory De-duplication Technique in Virtual Machines

Rodney Owens
Department of SIS
UNC Charlotte
Charlotte, NC 28223
Email: rvowens@uncc.edu

Weichao Wang
Department of SIS
UNC Charlotte
Charlotte, NC 28223
Email: weichaowang@uncc.edu

***Abstract*—Because of intellectual property, user privacy, and several other reasons, many scientific and military projects choose to hide the information about the data sets that they are using for analysis and computation. Attackers have designed various mechanisms to compromise the operating system or database management system to steal such information. In this paper, we propose a non-interactive mechanism to identify the data sets in use in a cloud computing environment when the virtual machine (VM) hypervisors adopt the memory de-duplication technique. Specifically, when multiple memory pages with the same contents occupy only one physical page, their reading and writing access delay will demonstrate some special properties. We use the access delay of the memory pages that are unique to some specific data sets to derive out whether or not our VM instance is accessing the same data sets as the target of the attack. The experiment results on a widely used scientific analysis software package ParaView demonstrate the practicability of the attack. We also discuss the mechanisms to defend against such attacks.**

## I. Introduction

With the ever increasing computation capabilities, storage space, and network bandwidth, more and more scientific and military projects are starting to use very large data sets for analysis, computation, and decision making. For example, NASA's Earth Observing System Data and Information System (EOSDIS) can generate two terabytes of data each day. Because of the intellectual property, user privacy, and several other reasons, many of these projects choose to hide the information about what data sets they are using for analysis and computation, even when some of the data sets are public. For example, a survey conducted in 2009 [1] shows that out of 62 groups that are requested to share their data or data sources, only 24 groups comply. To get access to such information, attackers have designed various mechanisms to compromise the operating system and database management system. Different schemes have also been designed to defend against such attacks.

The proliferation of virtual machine platforms creates a new path for non-interactive identification of the data sets in use. In a VM hypervisor, multiple virtual machines share the same hardware resources. Although perfect isolation among VMs is required by design, researchers have identified several mechanisms to break such isolation through schemes such as side channels. For example, researchers find that the shared cache may become a side channel for the detection of the web traffic access rate or even keystrokes of the co-resident VM instances [2].

In this research we seek to investigate data set identification in the cloud computing environment when the virtual machine hypervisors enable the memory de-duplication functionalities (such as VMware ESX and ESXi [3] and Extended Xen [4]). The memory de-duplication technique takes advantage of the similarity among memory pages so that only a single copy and multiple handlers need to be preserved in the memory, as shown in Figure 1. Here $VM1$ and $VM2$ share two identical pages so they only occupy four physical memory pages. (Note that we have both inter- and intra-VM memory de-duplication.) Although this technique can reduce the memory footprint size of VMs, it will break their isolation and introduce new vulnerabilities such as non-interactive memory page identification. The objective of this paper is to explore the vulnerability by constructing concrete attacks on a widely used scientific visualization software package called ParaView [5], and investigate the mechanisms to defend against such attacks.

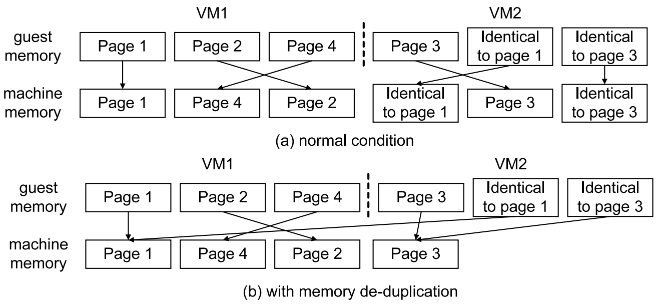


Fig. 1. Memory de-duplication reduces VM footprint size.

The overview of our approach is as follows. First, we will use the mechanisms in [2] to initiate a VM instance onto the same physical box as the target of the attack. Then we will read multiple data sets into the memory of our VM. Without losing generality, we assume that the target VM is using the data set $D_t$ for analysis and computation, and the data sets we open are $D_{a1}, D_{a2}, \cdots, D_{an}$. The objective is to determine whether or not $D_{ai}$ and $D_t$ contain many identical pages. If so, we will conclude that the target VM is actually using $D_{ai}$ for analysis. To achieve the goal, we will let the memory de-duplication mechanisms identify and merge those identical pages. Once this procedure is accomplished, we will introduce reading and writing operations to those memory pages that are unique to the data set $D_{ai}$. Since the hypervisor handles the operations differently for those de-duplicated pages and the pages with their own copies, we can measure the memory access delay to figure out whether or not $D_{ai}$ and $D_t$ are the same file.

The advantages of our approach are as follows. First and most importantly, it is a non-interactive data set identification procedure since during the attack we only conduct operations on our own VM instance. This non-interactive property will prevent the target VM from detecting the identification operations. Second, our experiments show that many scientific data sets, even when they are from the same broad field, contain a large number of unique memory pages. Reading/Writing operations to these pages can generate a measurable difference in access delay. Last but not least, our experiment results on ParaView show that this attack is practical.

The remainder of the paper is organized as follows. In Section II we describe the details of our data set identification approach. Section III presents our implementation and the experimental results when we use VMWare ESXi as the virtual machine hypervisor and ParaView as the data analysis software. We experiment with different combinations of guest OS and biological data sets. Section IV discusses the problems such as VM co-residence detection and prevention of the attacks. Finally, Section V concludes the paper.

## II. The Proposed Approach

### A. System Assumptions and Background

In the investigated scenario, we assume that an attacker can initiate VMs in the same cloud computing infrastructure as the target VM. We also assume that through the co-residence detection discussed in Section IV we can determine whether or not the target VM is running as a guest on the same host as the attacker's VM. The target VM may have very sophisticated Intrusion Detection/Prevention Systems in place. We assume that the attacker can submit queries to the target VM to initiate data access for analysis and computation. The attacker also holds some data sets and it wants to determine whether or not the target VM is using one of these data sets to resolve the queries.

We assume the attacker has root control over the VMs that it initiates. We also assume the attacker's VM has large enough memory (such as 512MB) to avoid very frequent page swapping. We do not assume the attacker can decide how many CPU cycles it is allowed to use, nor do we assume the attacker can decide how much physical RAM its VM is allowed to consume. These are reasonable assumptions based on current industry practice. Without losing generality, we assume that the host uses $4KB$ memory pages.

Since in our experiments we use VMWare ESXi as the hypervisor, below we briefly describe its memory de-duplication operations. A full description can be found at [6]. To avoid unnecessary delay during page loading, whenever a new page is loaded, ESXi will allocate a new physical page for it. Later, ESXi will use idle CPU cycles to locate the identical memory pages in physical RAM, and remove duplicates by leaving pointers for each VM to access the same memory blocks. While the reading operations to the de-duplicated pages will access the same copy, copy-on-write is used to prevent one VM from changing another VM's data. This procedure will incur extra overhead compared to writing to non-shared pages, which will lead to a measurable delay when a large number of shared pages are allocated and copied.

ParaView [7] is an open-source, multi-platform data analysis and visualization software package. ParaView allows users to generate visualizations to analyze their data using qualitative and quantitative techniques. ParaView is used by the organizations such as Army Research Laboratory, Sandia and Los Alamos National Laboratories, and NASA.

### B. Generation of Fingerprints of Large Data Sets

Considering the size of the data sets for scientific and military applications, their memory footprint often contains many pages. Some of these pages, however, cannot be used for data set identification since they are not unique for any specific file. For example, the header of the data files often follows a pre-defined format and contains information such as the dimension of the data, the number of records in the file, and the size of each data record. At the same time, the data files may also have some identical memory pages with the operating system or user applications. To determine whether or not a data file has enough unique memory pages, we propose to conduct off-line memory dumps of computers after they have loaded the file. The dump file is then cut into $4KB$ pages and compared to the memory pages of different operating systems and user applications. We adopt the mechanism in [4] and use hash results of the memory contents as indexes to locate the identical pages. Once we have categorized the memory pages of the data sets, we can find out which memory pages are unique to each data file. These coalesced memory pages will hereafter be referred to as *data file signatures*.

The data file signatures created in this fashion give us a real world representation to what can be found in the wild. The off-line memory dump and analysis is also much faster, easier, and only slightly less accurate than calculating what the similar memory pages would be based on the documented loading behavior of the data analysis software and the data files themselves. Based on these considerations, we believe that an attacker would most likely build data file signatures in the same fashion as we have for the proposed approach.

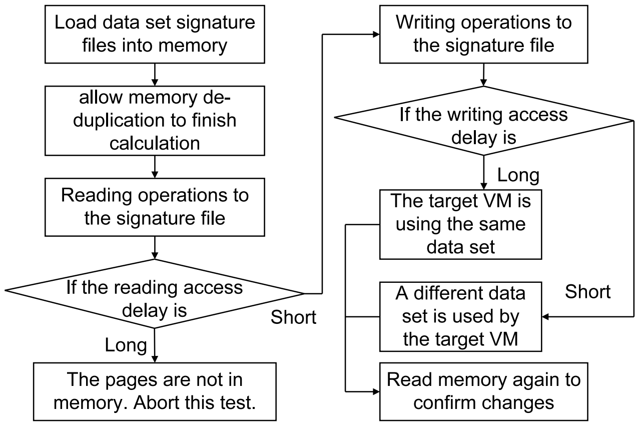
### C. Data Set Identification Procedures

As we describe in Section I, the virtual machine hypervisor ESXi uses different methods to handle the writing operations to the de-duplicated pages and pages with their own copies. For the pages with their own copies, the writing operation can be conducted immediately. For the de-duplicated pages, a new copy must be created first, introducing a delay. Our data set identification procedure, is to measure and detect the delay caused by the de-duplication between our data file signatures and the data sets in use at the target VM. To achieve the goal, we adopt the following schemes.

First, since we want to measure the extra processing delay caused by the writing operations to the de-duplicated pages of the data sets, we need to control at what time and to which pages such writing operations will be initiated. Fortunately, for many data analysis and visualization applications such as ParaView, the data sets are treated as read-only raw-data inputs. Therefore, we will compile the unique pages of each data set into single binary files for the attacker to load.

Second, since those long un-used pages may be swapped out by the hypervisor, we need to distinguish the delay of hard disk reading from that of copy-on-write. To accomplish this task, we plan to conduct a reading operation to the constructed data signatures before the writing operation. If the page is already in the memory, this reading operation can be accomplished immediately. On the contrary, if the page has been swapped out, this reading operation will force the hypervisor to execute a hard disk access with extra delay. Since ESXi will allocate a new memory page for the newly read content, the next step of writing will not provide us useful information. In this way, we can distinguish between the two types of delay.

With these basic components established, our data set identification procedure is illustrated in Figure 2.



Load data set signature files into memory
allow memory de-duplication to finish calculation
Reading operations to the signature file
If the reading access delay is
Long
Short
The pages are not in memory. Abort this test.
Writing operations to the signature file
If the writing access delay is
Long
Short
The target VM is using the same data set
A different data set is used by the target VM
Read memory again to confirm changes

Fig. 2. The proposed data set identification procedure.

When we confirm that our VM instance and the target VM are located on the same physical box through co-residence detection, we will submit queries to the target system to trigger the data access and analysis operations. We will then read the data set signature files into the memory of our VM. These files will be left alone to allow memory de-duplication algorithms to locate and merge the identical pages. Once we are sure that the de-duplication procedure is accomplished, we will conduct a reading operation to the signature files. If the reading access delay matches the hard disk loading time, we will abort the data identification procedure since the newly loaded pages all have their own copies. Otherwise, we will conduct a writing operation to the signature files. Since we already know these pages are in memory, based on the delay of the writing operation, we can determine whether or not they experience the copy-on-write procedure. If so, we know that another data file matching to this signature exists in the physical box.

## III. Implementation and Experimental Results

Although the basic idea of the proposed approach is straightforward, many issues need to be solved when we implement the attacks. For example, we need to examine the size of the data set signatures to make sure that the accumulated difference in delay is actually measurable. At the same time, we need to examine the timekeeping schemes in hypervisors so that we can measure the real delay. In this section, we present the details of our implementation of the attack and the experiment results.

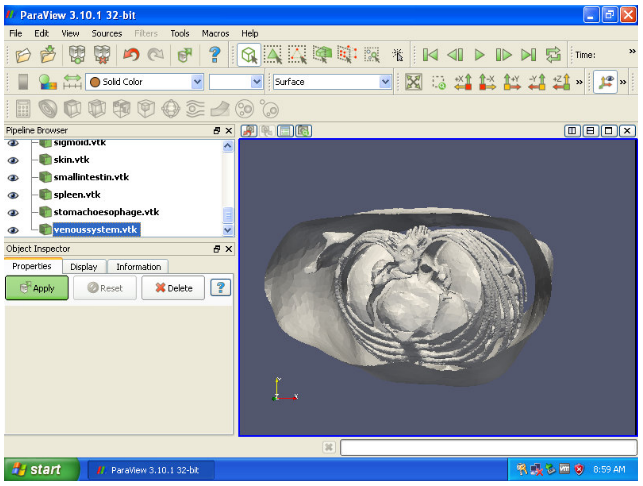


Fig. 3. Screenshot of 3D-IRCADb2.2, as used in experimentation.

### A. Experiment Environment Setup

Our VMWare ESXi server is running on a PC with a dual core 2.4GHz Xeon CPU, 4GB RAM, and SATA hard drives. We have chosen two Windows Operating Systems as the guest OSes for the target VM: Windows 7 32-bit, and Windows XP SP3. The attackers VM is Windows 95 because it has a smaller memory footprint than modern Windows Operating Systems. Here, having a small memory footprint prevents our attackers' VM from requiring excessive amounts of memory for loading memory page samples. We have chosen five biological data sets available from 3D-IRCADb [8] to serve as the data files for fingerprinting and identification. A screenshot of an example data set is available in Figure 3. We choose the five data sets available that have a signature of over 3,000 memory pages. Their basic information is shown in Table I.

| file & size | content | source | signature size |
|---|---|---|---|
| 3D-IRCADb1.1 (6.9MB) | liver tumor structure | 3D-IRCADb database | 3524 pages |
| 3D-IRCADb1.3 (13.4MB) | liver tumor structure | 3D-IRCADb database | 7836 pages |
| 3D-IRCADb1.5 (7.1MB) | liver tumor structure | 3D-IRCADb database | 3640 pages |
| 3D-IRCADb1.6 (7.2MB) | liver tumor structure | 3D-IRCADb database | 3661 pages |
| 3D-IRCADb2.2 (13.1MB) | Chest/Abdomen 3D CT-scan | 3D-IRCADb database | 7435 pages |

TABLE I
Selected data sets.

In order to build our data set signatures, we have generated and examined the memory dump files under different scenarios. Specifically, we need to examine the memory page contents under two conditions: (1) ParaView is initiated but no data file is opened; and (2) ParaView has opened the data file and generated the visualization. We experiment with different types of guest OS to investigate their impacts on the size of the signatures. The number of unique pages of each data set is summarized in Table I. Please note that in this table, we have cross-compared all the memory pages of the two guest operating systems, the memory pages of ParaView opened under different guest OS, and the pages of the data sets. From the table, we find that the signature files usually have the size of several thousands pages. The later experiment results will show that the accumulated delay of accessing these pages can be easily detected. At the same time, we find that when ParaView opens the same data file in different guest OSes, the memory dumps have many duplicate pages. This shows that ParaView is almost OS-independent. This property will definitely promote its wide adoption. However, it will also provide convenience to attackers in data set identification since the malicious parties do not need to first figure out the guest OS of the target virtual machine.

Another issue that we are facing is the accuracy of time measurement. Traditionally an operating system provides three methods to measure the length of a time duration: time of the day, CPU cycle counter, and APIC timer. The first method provides the granularity of seconds and it is too coarse for our application. The second method will be a good candidate for time measurement if the OS completely owns the hardware platform. In a VM-based system, however, it cannot accurately measure the time duration. For example, if a page fault happens during our reading operation, the hypervisor will pause the CPU cycle counter and switch to another VM. Therefore, the delay caused by hard disk reading will not be measured. Based on these observations, we choose to use the timestamp service provided by masm32 in winmm.lib to access the APIC timer. Specifically, we use the timeGetTime directive because it provides a 1 millisecond resolution. According to [9], VMWare has fully emulated the local APIC timer to provide accurate time readings, so the page fault handler built into ESXi to handle de-duplication will not pause the virtual local APIC timer. To further reduce extra delay caused by high level programming languages such as Java, we implemented the memory access and time measurement functions in assembly.

As illustrated in Figure 2, we have divided our program's functions into four groups of operations. The operation group one is to immediately read the first 32 bits of each data set signature's memory page and store the result of each read operation in the accumulator (EAX). Our program then sleeps the processor for a sufficient amount of time to allow de-duplication to occur. Our program then performs operation groups two through four immediately after each other. Group two does the same operation as group one, but reads the signatures after the sleep period. Group three writes junk data to the first 32 bits of every memory page for each data set signature. Group four then reads back the memory pages to confirm the changes.

## B. Experiment Results

We conduct four groups of experiments to evaluate the data set identification capability of the proposed approach under different levels of computation and memory access workload. We have two virtual machines running on the physical box. The operating system of the target virtual machine is Windows XP SP3. It has the latest version of ParaView installed. The attacker's VM uses Windows 95 as the operating system. The target VM has 512 $MB$ memory and the attacker's VM has 256 $MB$ memory. Since we have constructed the data set signature files in the binary format, we do not need to install ParaView on the attacker's VM. This is actually preferred, because writing junk data to memory pages in actual use by ParaView may cause the program to crash, which may be a detectable event on the hypervisor. We assume that the attacker has a rough idea of what sample data sets might be used by the target VM. In our experiments, since we have direct access to the target VM, we will activate ParaView to operate on a single selected data set at a time. Since the attackers do not know which data sets the target VM is using, we will load the signature files of all data sets into memory and conduct the reading/writing operations on each of them. Our results are shown in Figures 4 through 7. In addition to our 4 groups of operations, we first show in each figure the hard disk delay required to load each signature from disk. Since the access delays span across multiple degrees of magnitude, we use log-scale Y-axis. Since the signature files of different data sets have different sizes, we illustrate the average reading and writing time per memory page in the figures. Each node in the figures are the average value of five experiment runs with the same configuration. All time delays are measured in milliseconds. To help readers better understand the identification results, the page access delay of the data set read by ParaView on the target VM is always represented by "x".

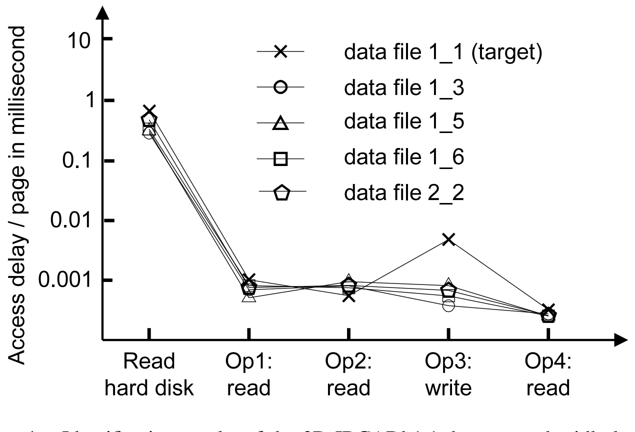


Fig. 4. Identification results of the 3D-IRCADb1.1 data set under idle level computation workload.

In the first experiment, we set up a baseline test case. Here the ParaView software on the target VM reads the 3D-IRCADb1.1 data set and generates the visualization. We choose this data set as the test case since it contains the smallest number of signature pages that still fit our 3000 page count lower limit. As shown in Figure 4, the hard disk delay is long. Next, the first read operation is short because the pages have just been freshly loaded into memory. After that, the target VM and the attacker's VM are left idle to give the de-duplication algorithms enough time to scan the memory. Since each VM has enough memory to store the signature files, we do not expect a lot of page swapping to happen. This is confirmed by the very short access delay of the second group of reading operations. The access delay of the writing operations, however, demonstrates the difference among the signature files. Here the delay of the signature file of the 3D-IRCADb1.1 data set is about twelve times longer than those of other data sets because of the copy-on-write operations. Our approach can successfully identify the data set in use in this baseline setup.

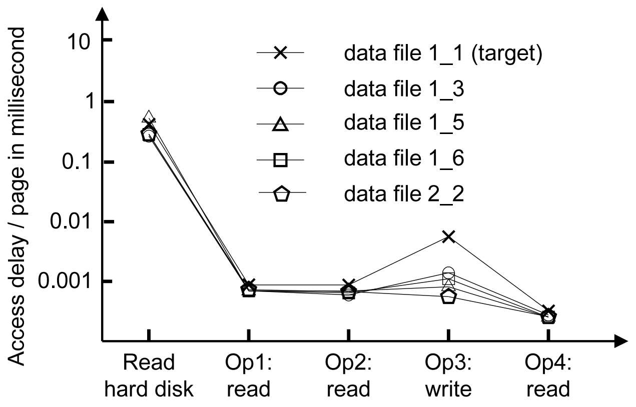


Fig. 5. Identification results of the 3D-IRCADb1.1 data set under moderate computation workload.

For the second experiment we continued with the 3D-IRCADb1.1 data set. This time, we also startup a Windows 7 VM and allow it to run along side the target and attacker VMs. We repeatedly ran the System File Checker (SFC) included with Windows 7 to simulate a normal moderate computer load by a third party Virtual Machine running on the hypervisor. The physical machine maintained fluctuating demands on the CPU, memory, and hard disk through the entire test. The results are shown in Figure 5. From the figure, we find that the reading and writing delays are very similar to Figure 4. The measured writing delay is still much longer than those of other data sets (4 to 6 times longer). From this figure, we find that the proposed approach will work properly when the signature file is as small as three thousand pages under normal real-world operating conditions.

In the third experiment, we want to assess what the impacts are of using a data set with a larger signature, so we used the 3D-IRCADb1.3 data set. To introduce the impacts of normal CPU, memory and disk usage operations, we again run the SFC included with Windows 7 in a loop during the experiment. As we describe above, the ParaView software on the target VM reads the 3D-IRCADb1.3 data set. After the visualization is generated, we leave the ParaView application alone to avoid extra computation and memory access overhead caused by the software. The results are shown in Figure 6. The write delay measured is still 2 times longer than those of other data sets. This figure shows that our approach can work properly with larger signature data sets under normal operating conditions.

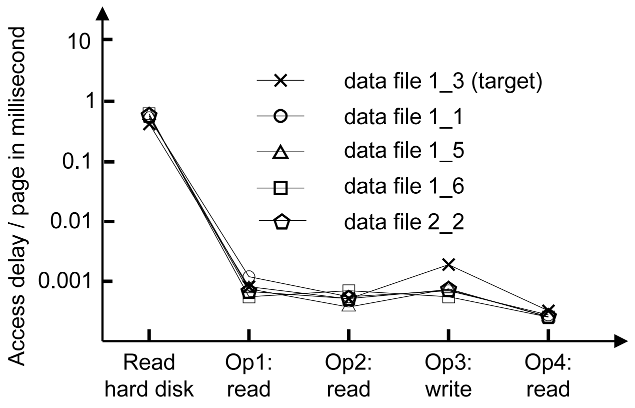


Fig. 6. Identification results of the 3D-IRCADb1.3 data set under moderate computation workload.

In the fourth experiment, we write a script to continually change the viewing angle on the target VM. The view point rotated between each viewing axis and back again within one second, causing the virtual CPU to peak out at 100 percent usage, with each physical core maintaining about 50 percent usage. We use the 3D-IRCADb1.1 data set to see if a smaller signature will have a noticeable access delay. The results are shown in Figure 7. The identification of the data set in use in the target VM can still be identified with a 10 times longer writing delay.

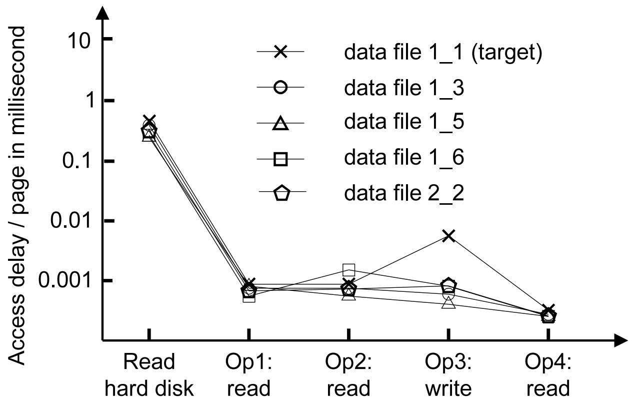


Fig. 7. Identification results of 3D-IRCADb1.1 data set under high level computation workload and memory demand.

## IV. Discussion

### A. The problem of VM instance co-residence detection

Although the experimental results in Section III are very encouraging, one problem is left unsolved: how can we put our VM instance onto the same physical box as the target and determine their co-residence. We plan to experiment with two adversarial strategies to place the attacker's VM onto the same physical box as the target. The first strategy is brute-forcing placement in which we will launch numerous instances over a period of time and conduct co-residence test discussed below. Previous research [2] shows that this simple approach has about 10% probability to successfully put the attacker's VMs onto the same physical box as the target. In the second attack strategy, we plan to explore the strong sequential and parallel placement locality of VM instances that have been shown in third party clouds such as Amazon EC2 [2]. With this property, if attackers launch VM instances relatively soon after the launch of the target, it has a better chance to achieve co-existence.

We understand that different VM management systems have different mapping policies among the virtual machines and physical boxes. For example, some systems use static mapping between the two groups. Under this condition, we can use the information such as the Dom0 IP addresses and internal IP addresses in the cloud to determine whether or not two instances are on the same physical box.

We plan to use two groups of mechanisms to verify co-residence of the attacker's VM and the target. In the first group we will examine the similarity of their Dom0 IP addresses and internal IP addresses in the cloud since many third party cloud management systems use static mapping between the addresses of VMs and the physical boxes. In the second group we will investigate load-based co-residence detection schemes [2]. The basic idea is to induce different levels of computation and data access loads onto the target and measure the operation delay of the attacker's VM instance. If the two sequences of events match very well, the two VMs have a good chance to be located on the same physical box.

### B. Prevention of the de-duplication based data set identification

Since the proposed data set identification mechanism uses the access delay to the memory pages to identify their contents, the defense mechanisms need to hide the difference. Two approaches can be used to achieve the goal. In the first approach, we can combine the unique pages of different data sets to construct a data file. When a VM instance tries to defend against the fingerprinting attack through memory de-duplication, it can periodically read the data file into its memory. In this way, the instance will demonstrate the signatures of multiple data sets and raise the difficulty level of identification. Please note that this approach can only defer the attack but not totally disable it since the real data set in use will still be in memory.

In the second approach, we can change the organization of the data records in the memory when they are loaded from the hard disk. In this mechanism, we can adopt different indexing schemes to organize the data records in the main memory. Therefore, even when the same data file is read at different VMs, the memory pages will still have different contents. The flexibility in data organization will not introduce too much overhead in information processing and analysis since many software packages such as MS SQL server are designed to support multiple logical equivalent plans for query processing [10].

## V. Conclusion

In this paper we propose a new data set identification mechanism for VM instances on hypervisors that enable the memory de-duplication functionality. The analysis shows that the reading and writing operation delay of the memory pages will demonstrate a measurable difference when they do not have their own copies in the memory. Experimental results on multiple biological data sets show that each of these sets contains a large number of unique pages that can be used as its signature. We can use the co-residence detection schemes to launch VM instances onto the same physical box as the target and determine the data set in use. Our approach has the non-interactive property and is more difficult to detect by IDS or network traffic monitor.

Immediate extensions to our approach consist of the following aspects. First, we plan to experiment with more types of operating systems and data analysis software to determine whether or not such an attack can be used to identify different data sets. We will also experiment with other hypervisors such as extended Xen to generalize the attacks. Second, we want to extensively study the designed prevention mechanisms. We will determine the frequency to access the data file that contains the unique pages of different data sets under various loads and assess the probability to fool the identification procedure. The research will provide guidelines for determining the tradeoff between memory management strategies and security in hypervisors.

## VI. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. OISE-0730065.

## References

[1] B. McCullough and R. McKitrick, "Check the numbers: The case for due diligence in policy formation," the Fraser Institute, February, pp.2–43, 2009.

[2] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proc. of ACM CCS*, 2009.

[3] VMWare, "Esxi configuration guide," VMware vSphere 4.1 Documentation, 2010.

[4] D. Gupta, S. Lee, M. Vrable, S. Savage, A. Snoeren, G. Varghese, G. Voelker, and A. Vahdat, "Difference engine: harnessing memory redundancy in virtual machines," *Commun. ACM*, vol. 53, no. 10, pp. 85–93, 2010.

[5] A. Cedilnik, B. Geveci, K. Moreland, J. Ahrens, and J. Favre, "Remote large data visualization in the paraview framework," in *Eurographics Parallel Graphics and Visualization*, 2006, pp. 162–170.

[6] VMWare, "Understanding memory resource management in vmware esx 4.1," VMware vSphere 4.1 Documentation, 2010.

[7] Kitware, "Paraview," http://paraview.org/, 2011.

[8] European Institute of Tele-Surgery - Institut de Recherche Contre les Cancers de L'Appareil Digestif, "Ircad/eits laparoscopic center," http://www.ircad.fr/softwares/3Dircadb/3Dircadb.php, 2011.

[9] VMWare, "Timekeeping in vmware virtual machines," http://www.vmware.com/files/pdf/Timekeeping-In-VirtualMachines.pdf, 2010.

[10] R. Rankins, P. Jensen, P. Bertucci, C. Gallelli, and A. Silverstein, *Microsoft SQL Server 2000 Unleashed.* Sams, 2001.