CS554 KVM-KSM Project Report

Eddie C. Davis

4-29-2018

I. Introduction

The Kernel-based Virtual Machine (KVM) is a loadable Linux kernel module that allows the operating system to act as a Type-2 hypervisor, performing hosting duties on top of bare-metal hardware without modifying the guest OS. The CPU must support hardware-assisted virtualization instructions such as Intel VT-X or AMD-V. Many virtual machines can be hosted on a single physical machine with this technology, motivating the need to reduce overall memory storage.

Kernel Same-page Merging (KSM) is a feature that enables a hypervisor such as KVM to merge the memory pages of different running virtual machines if the two share identical content. The process periodically scans main memory for duplicate pages and merges them into a single page mapped to multiple locations. The duplicated pages are marked as copy-on-write (COW) so a copy will automatically be made to separate the pages when one process modifies a shared page.

A covert channel is a computer security vulnerability that creates an opportunity for two entities to transfer information that should not be communicated, in violation of the security policy. Virtual machines running on a cloud computing service such as Amazon Web Services (AWS) or Google Compute Engine (GCE) are the entities in this case, and KSM is the medium through which the covert channel is formed.

The KSM process, *ksmd*, runs over a period of milliseconds (a configurable parameter), and scans memory for duplicate pages. The copy-on-write technique that indicates a shared page has been modified also introduces overhead in the form of extra CPU cycles. This deduplication overhead is measurable and can be used to construct a covert channel. The channel is constructed as follows.

The sender and receiver each load a certain amount of identical data into memory, e.g., by reading the same file. The sender, e.g., VM1, encodes the message by writing to certain pages. The shared pages written by the sender denote a 1, and those that remain identical a 0. The sender and receiver processes sleep on their respective virtual machines, allowing the KSM process to merge the identical pages. The receiver, e.g., VM2, then writes all of the shared pages and records the clock time required to write each page. The relatively brief access times indicate that the page has already been deduplicated by the sender, and therefore indicates a 1. Comparatively long access times indicate the shared page has been deduplicated by the receiver, and indicates a 0. Thus, the length of the message is constrained by the number of pages occupied by the identical information loaded into memory in the first step.

II. Implementation

The covert channel construction via kernel same-page merging technique was implemented as both a user space program (*memdupe*) and a kernel module (*kmemdupe*) in the C programing language. The program can run in one of three roles, a simple virtualization detector (*role*=0), as the *sender* of a message through the channel (*role*=1), or as the receiver (*role*=2). The number of seconds to sleep can also be specified on the command line, but defaults to five seconds. The third argument is the file to read into memory, but defaults to the *vim.tiny* binary as it is likely to exist on even lightweight Linux distributions. The size of the file is about 1.1 MB and occupies 259 pages at the default page size of 4 KB. The final argument is the KSM threshold. This is the value at which the write time ratio is sufficiently long to indicate that copy-on-write has occurred when writing shared pages and that the process is running on a virtual machine with KSM enabled on the hypervisor. The default threshold is 3X.

The *memdupe* program first attempts to determine whether it is running in a virtualized environment by checking for the existence of the *hypervisor* flag in the */proc/cpuinfo* file. This is for logging and evaluation purposes. Then the CPL register is read to ensure that the program is running in the correct privilege level, i.e., the user-space version is running at level 3, and the kernel module version at level 0. Then depending on the specified role, the program performs the steps as described in the introduction.

The file is loaded into memory using a buffer allocated with the *mmap* function ensuring the data will be aligned on page boundaries, and set to be mergeable with the *madvise* function. If the *readtwice* flag is set (and is by default), the file will be read twice to guarantee that the redundant pages will exist in memory for both sender and receiver.

Next the pages are written according to the role. The sender encodes the message to be transmitted through the covert channel. The default message is a simple “Hello!”. Each byte is translated into eight bits, and pages corresponding to each one bit are written to force deduplication of those pages. All processes then sleep for the specified number of seconds. All but the sender will write the pages again to test measure the time required to write each of the pages. Clock times were measured with the *clock_gettime* function. The receiver will build a bit string using the measured times, long access times indicating that the page was not written by the sender, and is a zero, short times were already deduplicated by the sender, and so denote one bits.

Finally, the tester will indicate whether the time to write exceeded the specified KSM threshold and therefore implied the presence of virtualization, the receiver will output what it perceives to be the decoded message (i.e., “Hello!”), and the sender will simply exit.

III. Evaluation

The technique was evaluated by measuring page write times in four scenarios, 1) running in the guest OS with KSM enabled on the host, 2) running in the guest OS with KSM disabled on the host, 3) running on the host OS with KSM enabled, and 4) running on the host OS with KSM disabled. Each condition was measured both loading the file only once, and loading the file twice.

The experiments were conducted on an x86-64 machine with a four core Intel Core i5-3450 CPU at 3.10GHz, 16 GB of DDR3 RAM, 64 K of L1 cache and 256 K of L2 cache per core, 6 MB of shared L3 cache, and four NUMA regions. The host operating system is Ubuntu Server version 16.04, booted with Linux kernel version 4.13.0-38.

The virtual machine guests are running in full virtualization mode on the KVM hypervisor using VT-x. Each VM is assigned is single threaded and assigned to one core. The virtual CPU has 64 K of L1 cache, 4 MB of L2 cache, and 2 GB of virtual DRAM. The guest operating systems are running the same Linux platform (Ubuntu) and kernel versions.

The *ksmd* daemon is running on both host and guests, but KSM is disabled on the guests (i.e., the */sys/kernel/mm/ksm/run* file contains 0). The experiments were run with two identical and newly booted virtual machines running (VM1 and VM2), with a sleep time of 60 seconds to allow KSM to merge duplicate pages, and reading the default file */usr/bin/vim-tiny*.

The results are summarized in Figures 1-4. Each data point is the mean clock time difference of five experimental runs. Figure 1 represents page write access times on the guest with KSM enabled, Figure 2 data were also measured on the guest but with KSM disabled on the host. Figure 3 contains data collected on the host with KSM enabled, and Figure 4 on the hist with KSM disabled.

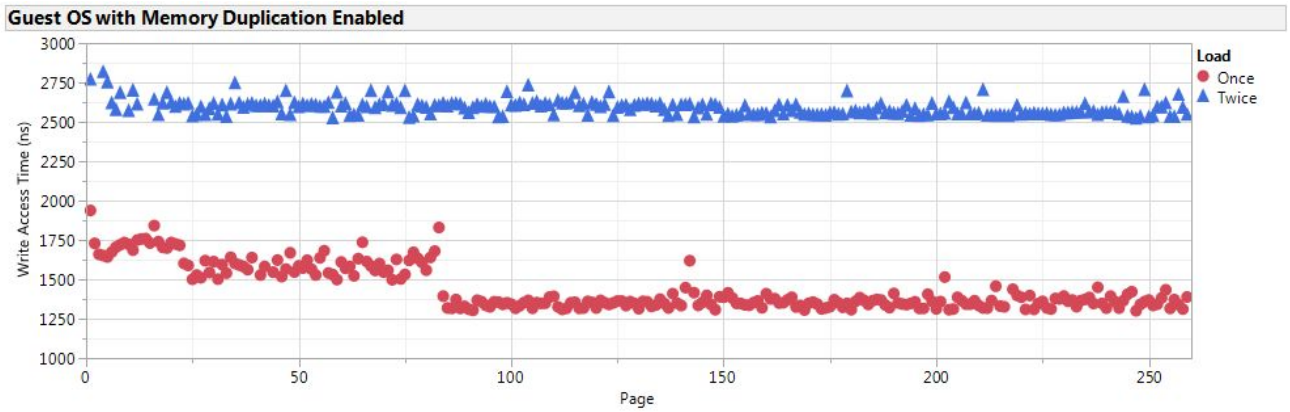

Figure 1: Guest OS with memory duplication enabled on the host (KSM is on).

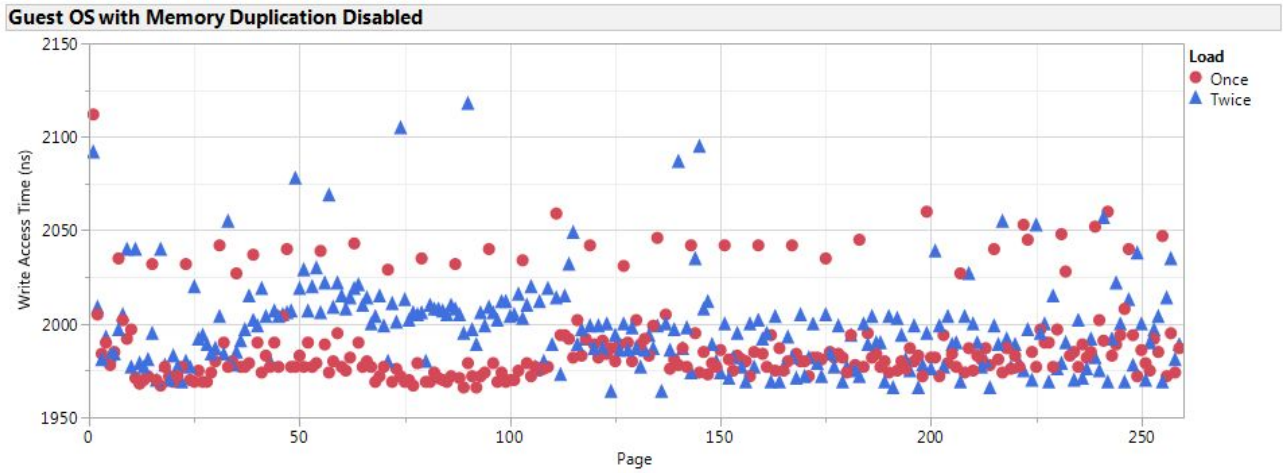

Figure 2: Guest OS with memory duplication disabled on the host (KSM is off).

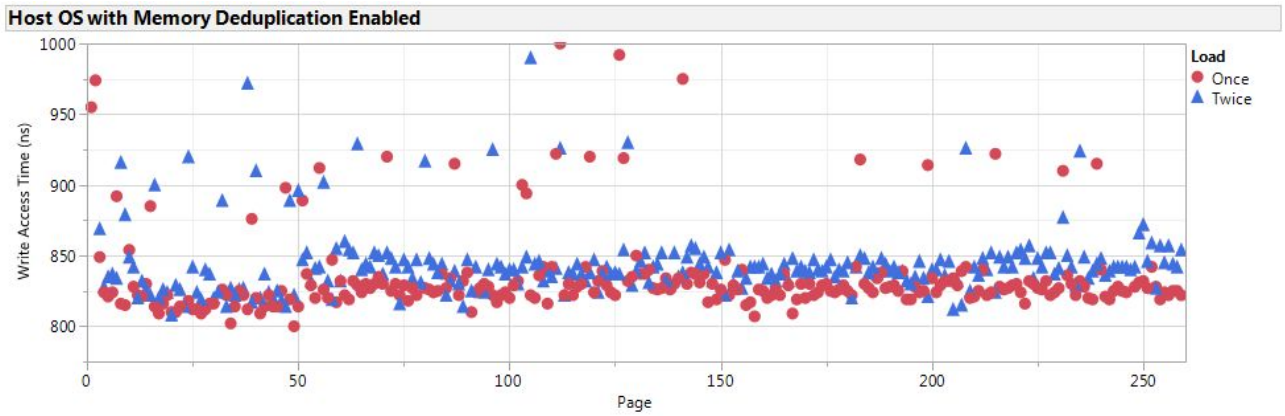

Figure 3: Host OS with memory duplication enabled (KSM is on).

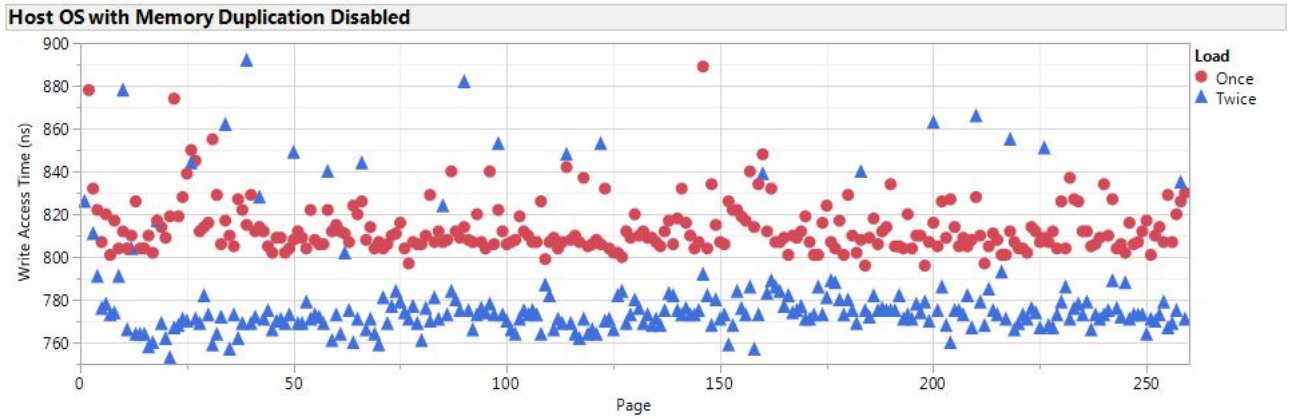

Figure 4: Host OS with memory duplication disabled (KSM is off).

IV. Conclusion

This project is an implementation of the covert channel construction between two virtual machines as described in Xiao 2013. The experimental results are largely consistent with those in the original paper. There is a consistent difference between write execution times if loading the file once compared to twice on the guest operating system with memory duplication enabled on the host operating system (hypervisor). However, the difference is not as pronounced as in the paper, approximately two times longer instead of up to six times. These results are visualized in Figure 1.

Executing the code on the guest with memory duplication disabled (Figure 2) or on the host with KSM enabled (Figure 3) exhibit little difference between loading the file once or twice. These results are expected. Write access times are generally longer on the guest OS, but this is also consistent with the results in the paper. There is a systematic shift between the read once and read twice write times on the host OS with memory duplication disabled (Figure 4), However, the difference is quite small in absolute terms, less than 100 ns, so is not likely significant.

The prevalence of virtualization in cloud based computing platforms has only continued to grow in the five years since this research was published. This work demonstrates the KSM memory duplication vulnerability has not been corrected as of the 4.13 version of the Linux kernel. As additional computing capacity is pushed to the cloud, such security concerns will become increasingly important.