Paper summary

ReVirt is useful for viewing and reconstructing an attack by replaying the complete, instruction-by-instruction of the virtual machine. The issues with the current logging system are that if the kernel gets compromised, the auditing logs also get compromised. Also, logs do not contain sufficient information to recreate the whole attack for the non-deterministic scenario. The ReVirt module lies between the host OS and the VMM kernel module to protect the integrity and completeness of the system even if the "villain" attacks the guest kernel or boot block. It logs enough information to replay the whole attack. ReVirt utilizes checkpointing, logging, and roll-forward recovery to achieve this. The VMM is smaller and provides a narrow interface; it can mostly contain the invasion of the villain hence making it a trusted computing base(TCB). UMLinux is used as the virtual machine which exports a similar interface to the host hardware to the guest OS. The VMM is a loadable module in the host Linux kernel plus tools to invoke the VMM module. Any system call which produces non-deterministic results is logged. The ReVirt module adds reasonable time and space overhead. Overheads due to virtualization are 13-58% for kernel-intensive workloads, and logging adds 0-8% overhead. Hence, ReVirt is a handy sandboxing mechanism for testing purposes.

Strengths

1. Adopts the fault tolerance techniques to replay long-term instruction. (use a good idea again)

2. Clear motivation and benchmarks.

Weaknesses

1. Only UMlinux guest OS is discussed. Are other OS's supported?

2. The kernel-intensive workload can have an overhead of up to 58%!

3. Checkpointing overhead is not discussed.

4. Host OS is modified to support ReVirt.

5. Some evaluation on identifying the attack by Revirt should have been explored.

Comments for author

1. Why does a narrow interface in the VMM mean more trustworthiness?

2. Instead of having ReVirt as a layer, can ReVirt be used as a utility driver as done in ballooning? What would be the tradeoff in such cases?

3. What if ReVirt is compromised along with the host OS during testing by the "villain"? Is there a mechanism to prevent it?