1 TwinVisor: Hardware-isolated Confidential Virtual Machines for ARM

TwinVisor is the first system that enables the hardware-enforced isolation of confidential VM's on ARM platforms. It makes use of two hypervisors, one in het normal world and the other in the secure world.

1.1 Introduction

For IaaS (Infrastructure as a Service) it is desirable that there is minimal intrusiveness into the tenants workloads, also clear security bounds should be easy to implement and enforce. ARM is working on solutions but ARM TrustZone is a mature technology which may be able to achieve the same. TwinVisor reuses existing hypervisors for the normal world but creates a small hypervisor for the secure world to focus on securing confidential VMs. The TrustZone hardware design poses three challenges, TrustZone's secure world is no more privileged than the normal world, TrustZone application have fixed resource requirements, the conventional usage model assumes that there are little world switches. To overcome these challenges the following solutions have been come up with, the (horizontal) H-Trap, split contiguous memory allocator, fast switch facility.

1.2 Background

Various solutions have been released to guard cloud tenants. ARM is working on its CCA solution which introduces realms, which is another secured world with virtualization support.

1.3 Overview

The goals of the TwinVisor are security, efficiency and minimal modification. The N-visor manages hardware resources and provides services while the S-visor protects and guards them, non confident VM's don't have the protection from the S-visor.

The attacker has no physical access to the devices, this is assumed because software attacks happen far more frequently than hardware attacks on cloud platforms. Defense methods for Denial of Service and side channel attacks are out of scope because their solutions are orthogonal to the solutions described here.

1.4 Detailed Design

1.4.1 Logical Deprivileging Model

A technique called H-Trap (horizontal trap) is proposed, which slightly modifies the N-visor to logically deprivilege it. Any hypervisor or VM configuration cannot affect the S-VMs execution until the S-visor starts to run this VM, making the N-visor less privileged than the S-visor. A call gate including SMC instructions is provided by TwinVisor to enable the N-visor to proactively enter the S-visor. Some registers need to be shared with the N-visor, these are checked before resuming execution in the S-visor, the others are randomised before the N-visor gets control. When a page fault occurs the S-visor calls the N-visor to use its page fault handler to directly modify the page which is checked by the S-visor before execution of the S-VM.

1.4.2 Cooperative Management of Memory Resources

If memory of S-VM's is non consecutive, the S-visor will run out of memory fast. The N-visor is also not aware of dynamic adjustment of memory security. A Split Contiguous Memory Allocator allocates a large contiguous area of memory during boot time and leases parts of it to a buddy allocator during runtime. This Split CMA tries to keep as many secure memory pages as possible contiguous to eachother to make sure that the secure memory regions are sufficient. The normal and secure ends of the Split CMA work together to maximize memory utilization. If there is a page fault the N-visor allocates new memory for the S-VM, when the control is transferred to the secure world the secure end of the split CMA transforms these pages into secure ones. When the normal world uses up all its available memory it can get some secure memory from the secure world. This memory that is lend to the normal world needs to be consecutive and at the end of the secure memory range so the secure memory needs to be compacted by the CMA.

1.4.3 Efficient World Switch

A fast world switch is introduced to reduce the overhead of world switching, this is possible due to certain redundant operations of the regular world switch. Lots of registers don't need to be wiped because the two hypervisors don't use them anyways.

1.5 Implementation

The S-visor fully reuses the I/O mechanisms and device drivers of the N-visor. The kernel loading logic is also reused, after the S-VM starts to boot, the kernel image is loaded into memory within a fixed GPA range. TwinVisor uses shadow I/O rings and shadow DMA buffers to be transparent to S-VMs. When a frontend driver in an S-VM issues an I/O request, the S-visor copies the request from the S-VM to the I/O ring. Conversely, if the backend driver in the N-visor raises an I/O completion interrupt, the S-visor in the N-visor raises an I/O completion interrupt, the S-vosr synchronizes the I/O ring to the shadow one and redirects the interrupt to the S-VM.

1.6 Security Analysis and Evaluation

The achieved security properties are

- The firmware and the S-visor are trusted during the system's lifetime.
- The integrity of S-VMs' kernel images is enforced by the S-visor.
- Each S-VM's CPU register states are protected by the S-visor.
- Each S-VM's memory is isolated from other S-VMs and the normal world.
- Each S-VM's I/O data is protected by the S-visor.
- All data and the control flow of each S-VM are protected by the S-visor.

1.7 Performance Evaluation

1.8 Hardware Advice for Future ARM

1.9 Related Work

1.10 Conclusion