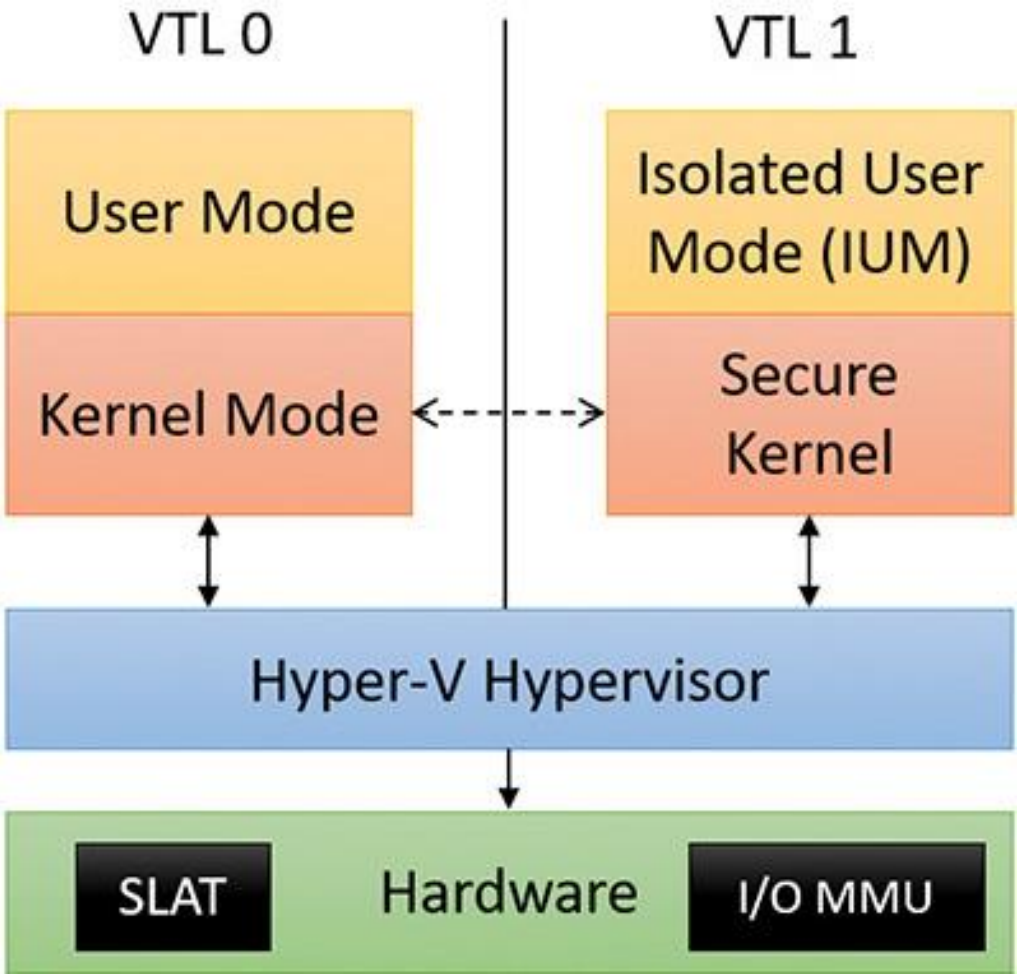
Virtualization-based security architecture overview

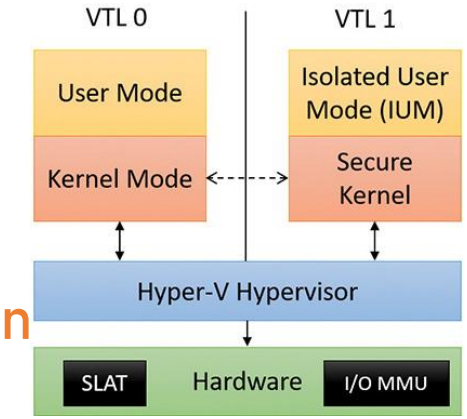
- Kernel-mode code has complete access to the entire system
- If an unwanted piece of kernel-mode code makes it into the system, the system is essentially compromised.
- Leveraging the hypervisor to provide additional guarantees against attacks, make up a set of virtualization-based security (VBS) capabilities, extending the processor's natural privilege-based separation through the introduction of Virtual Trust Levels (VTLs).

Virtualization-based security architecture overview



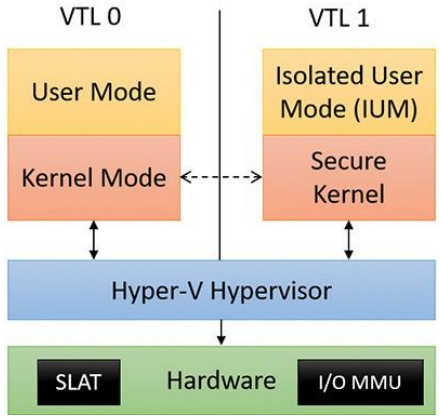
Virtualization-based security architecture overview

- The secure kernel is its own separate binary (securekernel.exe)
- Run-time user environment mode, called the Isolated User Mode (IUM)
- Restricts the allowed system calls that regular user-mode DLLs can make.
- Limiting which of DLLs can be loaded.
- A framework that adds special secure system calls that can execute only under VTL 1 (Iumdll.dll & Iumbase.dll).
- Sharing the same standard Win32 API libraries (reduction of the memory overhead of VTL 1)
- Copy-on-write mechanisms (prevent VTL 0 applications from making changes to binaries used by VTL 1)



Virtualization-based security architecture overview

- Privilege levels (user versus kernel) enforce power.
- VTLs enforce isolation.
- The secure kernel does not implement a full range of system capabilities (as proxy kernel).
- Protecting virtual memory by Second Level Address Translation (SLAT) (store secrets in locations)
- Protecting physical memory I/O Memory Management Unit (MMU) (virtualizes memory access for devices)



Trustlets

- Could an arbitrary user-mode process run in VTL 1?
- No, Just Trustlets!
- A special class of specially signed binaries
- Has a unique identifier and signature
- Secure kernel has hard-coded knowledge of which Trustlets have been created so far
- Impossible to create new Trustlets without access to the secure kernel