
Platform Security Summit 2019
A Renaissance of Trust:
Architecting the Hardened Access Terminal
(HAT)
Daniel P. Smith
Apertus Solutions, LLC
CC-BY-4.0: https://creativecommons.org/licenses/by/4.0/

Discussion for Today

- A quick historical review of trusted/secure computing and why everyone should care
- Explain where HAT started from and the initial groundwork
- Discuss some of the underlying concepts that have driven the HAT design
- Review the general architecture along with a notional implementation
- Close with the direction things are moving and how people can help

Privacy, Safety, and Trust

- Driven by a national debate on the effect of computer systems on privacy, the late 1960's and early 1970's saw a wealth of research on trusted computing
  - A culmination of that work can be seen in a pair of conferences held in 1973[1] and 1974[2]
  - The depth of critical thinking and the resulting concepts are the basis of security today
- In the early 1980's you began to see concepts like the Confidentiality, Integrity, Availability (CIA) triad take shape[3], all driven by the same threats and concerns we still face today
- Through the 1980's computers began to be used in safety critical situations and trusted computing was there to consider how to ensure safety[4]
- If one looks across all this work, it can be seen that the focus is on understanding trust and where it must be imparted to provide a degree of trustworthiness in a system

Relevance for Today

- Today more so than ever are,
  - Individual's personal/private information is being harvested by a vast number of entities, both government and private
  - Embedded systems and IoT are entrenched in everything from a power outlets and light bulbs to controlling the vehicles that moves around at vast speeds
- The fact is that a perfectly secure system does not necessarily mean that it will protect the user's privacy[1] nor will it guarantee that it will keep the user safe from performing an unintended function or experiencing fault during normal function[4]
- This does not make the converse true, an insecure system will not be able to provide the necessary guarantees that it can enforce its privacy or safety enforcement

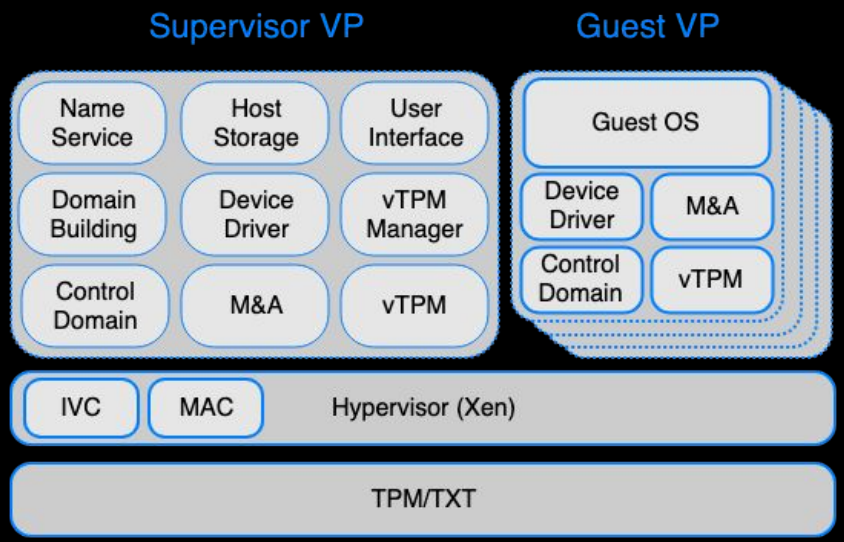
Call to Action

- It should no longer be considered acceptable for security to be an afterthought, bolt on the side feature or designed/developed as second class to operations
- As systems designers and builders it behoves to,
  - Grok what trust is and how eliminate reliance on implicit trust mechanism
  - Architect common, open security architectures
  - Design hardware to enable explicit trust mechanism usable by the security architectures

Introducing HAT

- At PSEC 2018 proposed the TrenchBoot project to develop a unified security architecture to provide meaningful integrity assertions about what software was loaded into the system
  - It is still in its infancy but has made huge strides to include inspiring the new kernel_info structure being introduced to Linux startup
- TrenchBoot is just the first piece of a larger secure computing architecture, today I am introducing the second building block, the HAT architecture
- HAT can be seen as an evolution of the OpenXT architecture with influences from the Secure Virtual Platform (SVP)[5] and MILS research

Secure Virtual Platform



- Began in 2002 as a study of how hardware virtualization could be leveraged to achieve advanced trust concepts and achieving higher assurance in platforms[5]
- Influencer for XenClient XT (OpenXT predecessor)
- Based on the VMM/VM isolation model to construct “Virtual Platforms”

MILS and Separation Kernels

- OpenXT, SVP, and most MILS solutions rely on a full featured hypervisor to function as the separation kernel
- A big issue with this approach is that if one looks at the thinking behind the separation kernel, and its predecessors the security kernel and reference monitor, it is supposed to only provide isolation and control the communication paths between the isolated components
  - As hardware become feature rich, their interfaces became rich
  - Hypervisors followed along in supporting these interfaces as to continue their intended role

Taking a Look at Today's Hypervisor

- Hypervisors generally have two primary functions
  - Management of the CPU virtualization
  - Management of physical machine emulation
- Hypervisors typically rely on an all-powerful management virtual machine
  - Xen - dom0, KVM – host Linux runtime, Hyper-V – host Windows OS
  - In trusted computing research, this VM is the equivalent of a Trusted Process
  - Unrestricted privilege and access to hypervisor and hardware
  - Responsible for hardware access management
  - Responsible for hardware emulation
    - Xen partially mitigates this with Stub Domains

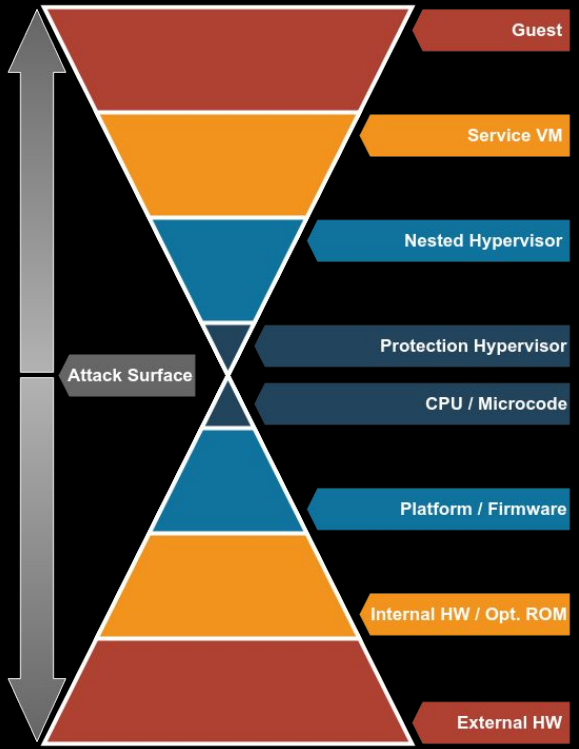
The Approach of Hypervisors Needs Rethinking

- The threat landscape for hypervisors has greatly increased
  - The pervasive use by cloud computing along with practically every desktop
    - Windows, Mac, BSD, and Linux all have a hypervisor embedded into them
  - Hypervisors have become fairly complex producing a larger attack surface
  - The result is an attack space that has breadth and depth
- Due to its position in the privilege rings, its compromise enables an attacker to exist outside of the protection model
  - There is work on enclaves in an attempt to mitigate/avoid this threat though I do not believe it can fully overcome it
- So what do we do,
  - Least Privilege tells exactly what we need to do, separate out system privilege

Hypervisor Privilege Separation

- The two primary functions must be separated from each other into a Protection Hypervisor (PX), sometimes referred to as a Level 0 (L0) hypervisor, and a Nested Hypervisor(s), sometimes referred to as the Level 1 (L1) hypervisor
- The functions of the all-privileged management VM needs to be separated and must become isolated peers to the nested hypervisors they support
- The context of the privilege separation should be driven by the disaggregation of the attack surface

Disaggregating the Attack Surface



- Due to the richness in hardware, hypervisors have become fairly complex producing a larger attack surface
- As a result hypervisors are now under attack, from above by the guest and from below by the firmware and hardware
- A Protection Hypervisor separates High Privilege operations along with risk laden system resources from the Nested Hypervisor, presenting a smaller Attack Surface

Protection Hypervisor

- The PX should strive to function as a separation kernel in the sense presented by Rushby[6]
  - This means it should be as small as possible and very limited in features, e.g. Rushby extolled that a separation kernel is logically simple enough that it should be implemented purely in assembly[7]
- The PX must enforce roles and privilege separation that partition the system and ensure communication only occurs across non-bypassable, always enforced policy controlled channels
- The PX must be self-protecting
  - It must retain just enough control over resources to ensure a component cannot leverage a resource to bypass the PX controls

Critical Function Isolation

- In today's implementations where critical functions are isolated, the mechanism is at the VM level
  - Often consisting of a general purpose operating system with a few limited cases where a special purpose operating system is used
- The PX will isolate critical functions such that dedicated “Single VM” nested hypervisor
  - A “Single VM” nested hypervisor is a shim capable of only running a single VM instance
  - The single VM may be a purpose built operating system (preferable) or adapted general purpose operating system
- The minimal critical functions that should be isolated are,
  - PCI for safely isolating hardware access
  - Network for mediated access to a shared hardware device
  - Storage for mediated access to a shared hardware device
  - Isolated multiplexing of the Roots of Trust for establishing platform integrity
- Depending on hardware and use cases then USB, GPUs, and other shared hardware resources

Hypervisor PCI Device Management

- PCI devices are core to modern computers but also expose a significant attack surface.
- The presence of IOMMUs often are thought to bring protection but even though the PCI-e Access Control Services (ACS) specification was introduced in 2015, its usage outside of server platforms is scarce at best for Intel
- For all major hypervisors, the solution for dealing with added complexity around the availability of ACS, or the lack thereof, has been to introduce more logic into the hypervisor to leverage its position in the privilege rings in an attempt to better control device access
- This introduces the risk that a malicious device may now leverage this increased attack surface between the devices and the hypervisor

PCI Isolation

- The PX must split the access to PCI bus between itself and a dedicated entity (PCI manager) outside the PX
  - The PX should mediate access to the bus using a limited set logic that is necessary to prevent the use of the PCI bus to attack the PX
- The PCI manager should be a single purpose kernel,
  - This reduces the potential attack surface and allows to optimize for performance
  - Should only contain PCI logic that is capable of,
    - Manage the physical bus
    - Function as a virtual IOMMU
    - Emulate a PCI bus to be presented to peer nested hypervisors
    - Emulate a PCI device for a peer nested hypervisor

Root of Trust Isolation

- While the TPM is not the only option for providing platform integrity related Roots of Trust, it does provide a concrete example in often misunderstood domain
- The TPM was designed more for an individual platform than a set of virtual platforms requiring their own Roots of Trust
- The physical TPM should be used to assert the integrity of the PX and should not be accessible by any other portion of the system
- The PX should provide the ability to have an isolated instantiation of one or more virtual TPMs (vTPMs)
    - Implementation must meet the shield requirements (shielded locations and protected capabilities) of a compliant TPM

Hardened Access Terminal

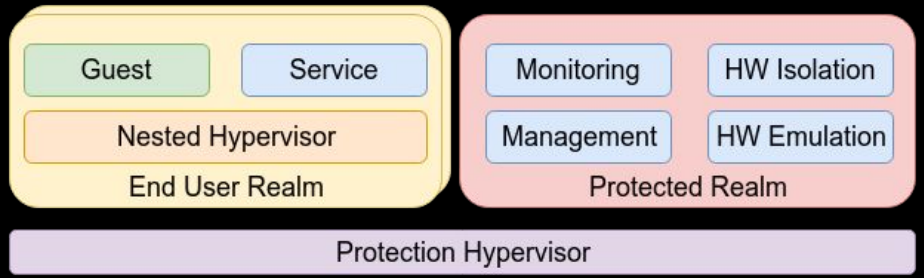
The HAT architecture is the culmination of integrating all these concepts

HAT Principles

- Essential Properties
  - Strong Isolation – The isolation provided by a solution must be explicit and verifiable
  - Verifiable Integrity – A solution must be able to assert explicit statements of any component's integrity
  - Fault Tolerant – A failure of a component should not degrade the Isolation or the Integrity of the platform
- Derived Properties
  - Data flows in the system must be Non-bypassable, Always Invoked, and Tamperproof
  - The Roots of Trust must provide explicit assertions
  - Components must be singular in their responsibility with minimal implementation complexity
  - Integrity statements must provide meaningful evidence regarding the properties of the component [8]
  - A components role/responsibilities and its interactions with other components must be represented in a policy that can be evaluated for completeness

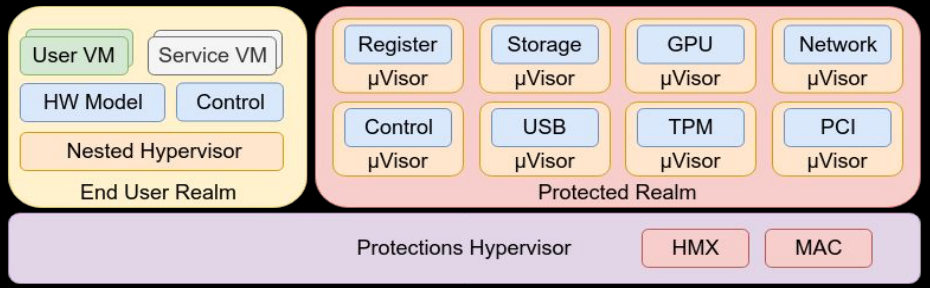
HAT Generalized

- PX divides the system into two realms, the Protected Realm and the End User Realm
- End User Realms are under the auspices of the Nested Hypervisor, free to participate or provide its own security architecture/policy
- End User Realms consume services/resource provided by the Protected Realm, as allowed by policy
- Protected Realm is a policy enforced isolation realm consisting of system resources provided as distributed computing services
- Protected Realm may contain entities of varying degrees of trust
- All data flows to/from an entity in the Protection Realm must be mediated by the PX



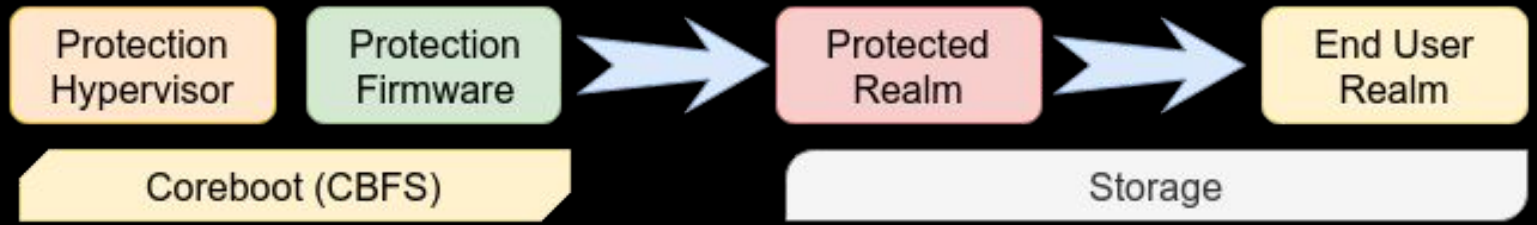
A Notional HAT Instance

- End User Realm(s) provides the user facing/exposed componentry
- Nested Hypervisor is responsible for securely isolating the guests and services hosted within an End User Realm
- Open Source or Commercial Hypervisor may be used as the Nested Hypervisor, with or without knowledge of the Protection Hypervisor
- Platform services and high risk components are isolated within a micro-visor
- A Register is used to lookup services
- Shared resources storage, network, and graphics are isolated and multiplexed
- Platform management is split with user facing Control service and isolated critical logic provided by the Control service in the Protected Realm



HAT Launch Architecture

- The PX must be instantiated from a strong Root of Trust with short trust chains, e.g. SRTM is good but DRTM would be better
- The PX should utilize a Protection Firmware that is designed to run directly on the PX
- The Protection Firmware is responsible for instantiating the implementation specific portions of the Protected Realm necessary to finish launching the system
- The Protection Firmware should, but not required, be evicted from memory
- Depicted below is an exemplar approach



From Theory to Reality

- TrenchBoot - standardizing launch integrity and exploring ways to re-establish integrity during runtime life cycle
- Xen – a few efforts being pursued
  - Using KCONFIG to create an instance of a Protection Hypervisor, pico-Xen (pX)
  - Introducing boot domain (domB) that would be focused on limiting compiled in privilege for the zeroth domain, equivalent of the “Protection Firmware”
  - Leveraging Argo’s upcoming labeled communications as policy enforced HMX
  - Proponent of moving away from dom ids for UUIDs and the use of a Name Service accessed via Argo to locate service domains
- Qubes/Apertus/ZEDEDA collaboration on a vTPM rooted in SGX for shielding requirements
  - Potential to demonstrate a hardware RoT without a physical TPM

Moving Forward

- How can one help
  - Hardware needs to continue to evolve to better and more securely support isolation
  - Collaborate on HAT mailing list, hat-devel@googlegroups.com
  - Contribute to Open Source projects that enable the HAT architecture
  - Fund Apertus Solutions, Qubes, and other entities that are involved in the realization of HAT

References

1. Davis, R. "Government looks at: privacy and security in computer systems." Computer and Business Equipment Manufacturers Association, Washington, DC (1973).
2. Renninger, Clark R., and Clark R. Renninger. Approaches to Privacy and Security in Computer Systems. 1974.
3. Jacks, E. L. "Computer Security Interest in the Private Sector." Proceedings of the Second Seminar on the DoD Computer Security Initiative Program.
4. Rushby, John. "Kernels for safety." Safe and Secure Computing Systems (1989): 210-220.
5. https://github.com/OpenXT/docs/blob/master/presentations/2016-06-07-openxt-summit/03%20-%20Loscocco%20-%20Virtual%20Platform%20Research.pdf
6. Rushby, John M. Design and verification of secure systems. Vol. 15. No. 5. ACM, 1981.
7. Rushby, John. "A trusted computing base for embedded systems." Proceedings 7th DoD/NBS Computer Security Conference. Citeseer, 1984.
8. Coker, George, et al. "Principles of remote attestation." International Journal of Information Security 10.2 (2011): 63-81