Commonality and Variability Analysis for Xenon Family of Separation Virtual Machine Monitors (CVAX)

JAMES KIRBY JR.
JOHN MCDERMOTT

Center for High Assurance Computer Systems
Information Technology Division

GRADY H. CAMPBELL JR.

Prosperity Heights Software
domain-specific.com
Annandale, Virginia

July 18, 2017

REPORT DOCUMENTATION PAGE

1. REPORT DATE (DD-MM-YYYY) 18-07-2017	2. REPORT TYPE Memorandum Report	3. DATES COVERED (From - To) September 2016 – October 2016

4. TITLE AND SUBTITLE

Commonality and Variability Analysis for Xenon Family of Separation Virtual Machine Monitors (CVAX)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

James Kirby Jr., John McDermott, and Grady H. Campbell Jr.*

5d. PROJECT NUMBER
55-8089-0M

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory, Code 5542
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/5542--17-9717

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Information Systems & Cyber Security
RD / ASD(R&E) / AT&L / Department of Defense
4800 Mark Center Drive, Suite 17C08
Alexandria, Virginia 22350-3600

10. SPONSOR / MONITOR'S ACRONYM(S)

OSD/OUSD/ATL/ASD(R&E)/RD

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
*Prosperity Heights Software, domain-specific.com, Annandale, VA 22003
Note: CLEARED For Open Publication, May 03, 2017, Department of Defense,
OFFICE OF PREPUBLICATION AND SECURITY REVIEW 17-S-1587.

14. ABSTRACT

The objective of this document is to define the composition of the Xenon Family of Separation Virtual Machine Monitors. This will support development of a Xenon family that can operate in a variety of computing environments, such as cloud computing, real-time systems, and mobile platforms. Further, through the Xenon Family, DoD can take advantage of improvements provided by the evolving open-source Xen hypervisor. The technical approach is a systematic application of Software Product Line Engineering (SPLE). A systematic application requires describing the family and how to produce desired family members.

15. SUBJECT TERMS

Separation virtual machine monitor	Software product line engineering
Xenon	Software family
Xen hypervisor	

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON James Kirby Jr.
a. REPORT Unclassified Unlimited	b. ABSTRACT Unclassified Unlimited	c. THIS PAGE Unclassified Unlimited	Unclassified Unlimited	11	19b. TELEPHONE NUMBER (include area code) (202) 767-3107

Xenon Family of Separation Virtual Machine Monitors

Xenon Commonality and Variability Analysis (XCVA)

Preface

The objective of this document is to define the composition of the Xenon Family of Separation Virtual Machine Monitors.[1] This will support development of a Xenon family that can operate in a variety of computing environments, such as cloud computing, real-time systems, and mobile platforms. Further, through the Xenon Family, DoD can take advantage of improvements provided by the evolving open-source Xen hypervisor. The technical approach is a **systematic** application of Software Product Line Engineering (SPLE). A systematic application requires describing the family and how to produce desired family members.[2]

SPLE moves software development away from one-size-fits-all deployment. This facilitates fine-grained customization to meet customer needs. Capabilities that a customer doesn't want, and their supporting infrastructure, can be easily removed by customers and subject matter experts who are not programmers. Such feature removal reduces software attack surface, facilitating cyber security. Eliminating unneeded features reduces memory footprint and processor utilization, increasing execution efficiency. Further, capabilities can be tailored to meet specific customer needs.

This living document describes the composition of the Xenon Family by providing two sets of assumptions. One set of assumptions, *Commonalities*, describes what all members of the family have in common. A second set of assumptions, *Variabilities*, describes how members of the family may differ. Commonalities and variabilities are expected to evolve as work on the Xenon Family progresses. Changes to this document will reflect that evolution.

Separately, a *decision model*, which comprises a formal definition of the identified variabilities, contributes to producing desired family members. *Resolving* decisions in the decision model can precisely distinguish a desired software member from others in the family. The decision model defines the range of adaptability required of the set of *adaptable components* from which family members are constructed. The adaptability of each component is represented by *parameters of variation* that control the derivation of each needed adapted component. The decision model provides a mapping from resolved decisions to values assigned to parameters of variation of a particular adaptable component, which describes the desired adapted component.

---

[1] McDermott, John, et al. "The Xenon separation VMM: Secure virtualization infrastructure for military clouds." *MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012*. IEEE, 2012.

[2] Campbell, Grady H. "Renewing the product line vision." In *Software Product Line Conference, 2008. SPLC'08. 12th International*, pp. 109-116. IEEE, 2008.

This document will lay the groundwork for subsequent efforts (proposed in *Software Product Line Engineering for Mission Critical Systems*) to develop a family of secure and robust VMMs (aka hypervisors) for the C4I community, and concurrently to refine and validate an approach to developing and sustaining such families for mission critical systems.

Domain Synopsis

The Xenon Virtual Machine Monitor (VMM) provides secure, sustainable virtualization technology required to take advantage of advanced hardware capability, mitigate hardware redundancy, and maximize resource utilization. Allowing a single computer to support multiple execution environments by virtualization software makes security separation an important issue. Sharing hardware introduces the possibility of interference between logically distinct execution environments, which can propagate faults and leak information. Rigorous separation guarantees that sharing hardware is undetectable to logically distinct execution environments.[3] While a Virtual Machine Monitor (VMM or hypervisor) allows one computer to support multiple execution environments, separation kernels represent the upper bound of practical software separation strength. But separation kernels require specialized hardware. They are not justifiable for complex commodity hardware and software. A *separation* VMM sufficiently relaxes the strict size and simplicity goals of a separation kernel to support commodity hardware and commodity guest operating systems.

The following sections of this document describe, in turn, the source of the Xenon VMM, commonalities of all members of the Xenon Family, and how members of the Xenon Family may differ from one another. The document concludes with a glossary of terms.

Source of Xenon Separation Virtual Machine Monitor

Xenon is constructed from Xen by:

- Replacing Xen's FLASK security architecture with the simpler Xenon MSM security architecture that provides a simpler, more intuitive interface for customers.

- Reducing in size and refactoring the source code to reduce significantly its complexity.

- Adding a network access policy capability, enforced by Open vSwitch, to restrict VM network access.

- Reducing the attack space of Dom0 and hardening it with SELinux and Linux Namespaces.

- Providing a local Graphical User Interface (GUI) to support security policy authoring, VM management, and VM resource consumption monitoring (CPU usage, network usage, memory usage, etc.).

---

[3] J. Rushby. Separation and Integration in MILS, TR SRI-CSL-08-XX, February 2008.

- Providing a remote GUI to manage and monitor multiple Xenon VMMs from a remote host.

Xenon Commonalities

The following is true of every member, x, of the Xenon Family of Separation Virtual Machine Monitors:

c1. x is a Type 1^4 "bare metal" separation VMM.

c2. x is constructed from the open source Xen VMM, with fewer features and a simpler implementation.

c3. x is designed to thwart specified threat models.

c4. x supports one Dom0, multiple user DomUs, and multiple service domains, some of which may be stub domains that run a minimal virtual machine.

c5.   The security features of x are always active.

c6. x requires Xenon's MSM security module.

c7.   Conflict sets restrict which domains of x execute simultaneously.

c8. x provides only scrubbed memory for any use.[5]

c9.   Security policy in x comprises hypercall policy, separation policy, and resource allocation policy.

c10.   Hypercall policy in x comprises:

    a. Hypercall permissions, which specify hypercalls a VM may make.

    b. The maximum rate at which the VM may make hypercalls. [6]

c11.   Some service domains in x may be trusted more than user domUs, but not as much as dom0.

c12. x determines which VMs are connected.

c13.   Strict separation in x by disallowing simultaneous execution of mutually-exclusive VMs.

c14. x logs VMM policy violations, e.g. hypercall policy violations.[7] Security logs are inaccessible to offending VMs.

c15. x security policy specifies how many security violations by a VM require halting its execution.

---

[4] Type 1 VMMs handle interrupts and virtualization-sensitive instructions directly. Type 2 VMMs delegate interrupt handling and sensitive instruction processing to a large conventional operating system.

[5] **Rationale**: to prevent leakage of information via reused memory.

[6] **Rationale:** to resist hypercall flooding attacks.

[7] Typical virtualization products only log externally visible security issues.

c16. Security administrator of x can halt executing VM and resume or destroy paused VM.

c17. x allows fine-grained modification of security policy without requiring rebuilding or rebooting VMM.[8]

c18. Any particular attacker can vary, including attacker's initial access, initial knowledge, and capabilities.

c19. Threats posed to x by a given attacker can vary, including what component(s) of x are threatened and what harmful outcome(s) are sought.

c20. Misuse cases an attacker employs can vary, including what flaws and errors in x the attacker exploits and their order.

c21. Versions of applications and OSs executing on each VM can vary.

Xenon Variabilities[9]

Variabilities characterize what customers want to customize, distinguishing the x they want from other members of the Xenon Family of Separation Virtual Machine Monitors. In the following, "can differ" describes what can distinguish one family member from another. In contrast, "can vary" describes what changes during execution of a particular family member.

v1. Threat models that x thwarts can differ.

v2. Changes that x will adopt from the evolving Xen code base can differ[10].

v3. Allowed hypercall rate may or may not be enforced.

v4. The response of x to a domain causing security violations may be fixed at VMM initialization or at domain initialization, or it can vary during execution of a domain.

v5. The Scheduler that x supports can differ, depending upon whether x is supporting an application in the cloud (implying a load-optimizing scheduler), a real-time application with a provably safe scheduler, a weapon system, a weapon support system, a mobile application, or a server application.

v6. The assurance to which x is built can differ, depending on the sponsor (e.g., military, intelligence community, other government, commercial, medical) and upon the type of system (e.g., application in the cloud, real-time application, weapon system, weapon support system, mobile application, or server application). Possible assurance standards include: Common Criteria, SP800-53, MIL_HDBK-516C.

v7. The computing environment in which x operates can differ:

---

[8] **Rationale**: to support scalability, reduce downtime and expensive VM migrations
[9] Binding time variabilities?
[10] This is very open-ended. We need to find a way to manage or limit it.

a. *x* can be deployed in the cloud to provide secure cloud operations.

b. *x* can be deployed to provide safe and secure partitioned real-time execution for weapon and related systems.

c. *x* can be deployed to provide secure partitioned execution for mobile devices with limited capabilities.

v8.   The processor on which *x* executes can differ. Possible supported processors include:

a. Intel X86

b. AMD X86

c. ARM

v9.   The Visual Management Interface (VMI) provided by *x* can differ. Some possibilities are:

a. *x* provides a security management VMI that allows security administrator to author security policy, manage network security and connectivity, and monitor VM resource consumption.

b. The *x* security management VMI provides default security policy and templates with example configurations.

c. There is no VMI; the *x* security policy specifies which pairs of VMs may communicate with one another.

d. The *x* security management VMI supports automatic configuration of separate device driver and network driver VMs.

e. A remote security management VMI provided by *x* allows remote security administrator to monitor and manage a Xenon enclave from a remote host.

v10.   The virtualization tool stack *x* implements can differ.

v11.   The Xenon Policy Tool provided by *x* can differ.

v12.   The IT features (e.g., Xen memory balloon driver, transcendent third-level memory sharing) can differ.

v13.   When security policy in *x* can change can differ. Some possibilities are:

a. Security policy can be changed dynamically, without rebooting the VMM.

b. Security policy is fixed at build time.

v14.   Specialized driver domains provided by *x* can differ.

v15.   *x* may comprise one or more physical hosts.

v16.   Whether *x* has automatic ***Load Balancing*** can differ.

v17.   Whether *x* supports ***Moving Target Defense*** can differ. If it is chosen, the following may apply.

a. VMs of *x* may migrate from one physical host to another.

b. Rules prescribing VM migration in *x* can differ.

c. A VM on one physical host of *x* may serve as hot standby for a VM on another physical host of *x*.

d. Rules prescribing when a VM may serve as hot standby for a VM on another host can differ.

e. Rules prescribing when and how the IP of a VM can vary can themselves differ.

v18. Whether *x* supports ***Situation Awareness*** can differ. If it is chosen, the following may apply.

a. *x* may collect security information from each physical host.

b. Rules prescribing what information is collected can differ.

   i. Information may be collected from sensors at VMs and network driver domains.

   ii. Information may characterize mission and assets

c. Rules prescribing when information is collected can differ.

d. Rules prescribing how collected information is processed can differ.

   i. The priority associated with each type of event collected can differ.

e. Where *x* sends collected information can differ.

   i. Whether *x* sends information collected to the security domain can differ.

   ii. Rules prescribing where to send collected information can differ.

v19. ***Continuity of Operations*** may be provided, allowing *x* to continue to operate without interruption while its individual VMs are patched and updated.

v20. Whether *x* supports networking among VMs or with the outside world can differ.[11]

Glossary

- **Attacker.** Threat agent. Characterized by

  o **Initial access.** Physical, geographical, and logical network location of attacker; security privileges and credentials possessed at start of attack.

  o **Initial knowledge.** Knowledge other than security credentials of, e.g., network architecture, configuration settings, source code, UUIDs.

  o **Capabilities.** Team size, talent, experience, available time, resources, etc.

---

[11] **Rationale**: Some real-time or weapon system members may not have network capabilities.

- o **Attacker potential.** Weak (can only use known vulnerabilities), low (can discover and use single fresh flaws, some coding skills), moderate (can discover and integrate multiple fresh flaws, skilled development team, dedicated infrastructure), high (distinguished from moderate attacker by academic and theory-based capabilities).

- **Conflict set.** Set of DomUs. DomUs in different conflict sets are not permitted to execute simultaneously.

- **Continuity of Operations.** VMM continues to operate while its VMs are patched and updated.

- **Domain**. Xenon's unit of resource allocation, scheduling, and separation that partitions computations. Comprises VCPUs, memory, and event channels. A domain implements a VM.

- **Domain0.** Also Dom0. The control domain, which has special control, device, and service privileges. Dom0 may be disaggregated to produce *service domains*. A disaggregated dom0 retains control capabilities but no longer provides device drivers or system services.

- **DomainU.** Also DomU, guest domain, User Domain. Domain without the special privileges of dom0.

- **Driver domain.** A stub domain that runs a device driver.[12]

- **Enclave.** Set of VMs and resources allowed to serve together.

- **Event channels.** Virtual interrupts that virtualize hardware interrupts and provide additional inter-VCPU communication.

- **Guest.** An operating system running in a domain. The guest is considered to be running on a VM.

- **Hot standby.** A VM that maintains identical state to another and is prepared to assume its execution on demand.

- **Hypercall.** Guests use hypercalls to communicate with VMM and request services. Hypercalls are analogous to system calls in an operating system. Some hypercalls are divided into several subcommands, just like system calls, to simplify the hypercall implementation.

- **Hypercall permission**. Permission to issue a particular hypercall.

- **Hypercall profile.** Set of hypercall permissions.

- **Load Balancing.** Virtualization load balancing relates to network load balancing, which redirects and allocates network traffic to multiple servers so that each server faces a similar network load. Virtualization load balancing allocates virtual machines to a set of physical host machines, so that each physical host machine has a relatively balanced load of virtual machines. Balancing is typically achieved

---

[12] Compromised driver domain leaves Dom0 unaffected

by migrating virtual machines from heavily loaded hosts to hosts with lighter loads**.**

- **Machine memory.** Xen's terminology for hardware memory present on a chip.

- **Misuse case.** Abuse case. Attacker-product interaction that the attacker exploits, e.g., flaws in hypercalls, errors in memory mapping, tampering with event channels to inject spurious virtual interrupts.

- **Moving Target Defense.** VMs may migrate among physical hosts, their IPs may vary, they may serve as hot standby for VM on another physical host.

- **MSM security module**. Checks hypercall permissions, limits security policy violations per domain, limits rate of hypercalls per domain, and enforces limits on connections between domains.

- **Physical memory.** Xen's terminology, short for pseudo-physical memory. Physical memory is the Xen term for the memory abstraction that VMM provides to guest operating systems which function as though they were implementing their own virtual memory.

- **Scrubbed memory** refers to memory that provides no indication of its previous contents.

- **Security tag.** Security tags express Xenon security policy with respect to separation. Xenon separates VMs with different tags .

- **Security policy.** "Succinct statement of a system's protection strategy." [13]

- **Separation kernel.** Strongest known means of software separation. Separation kernels are special VMMs that are small enough and simple enough to be mathematically verified. [14]

- **Separation VMM.** Best alternative to a strict separation kernel. A separation VMM relaxes the strict size and simplicity goals of a separation kernel far to support commodity hardware and guest operating systems. Because they address all of the features of commodity hardware, separation VMMs are too large for formal mathematical verification. However, separation VMMs are small enough and simple enough to be completely specified by semiformal means, i.e. they are smaller and simpler than conventional VMMs. A separation VMM has a complete systematic assurance argument that it isolates guest VMs from each other and strongly protects itself from tampering. A separation VMM provides the strongest separation of cloud VMs that is consistent with virtualizing complex commodity operating systems, on shared complex commodity hardware. [15]

- **Service domain.** A domain that performs a specialized utility service. A service domain of *x* is not used directly, but provides services to other virtual machine

---

[13] Anderson, Ross. *Security engineering*. John Wiley & Sons, 2008.

[14] McDermott, J., et al. "The Xenon separation VMM: Secure virtualization infrastructure for military clouds." *MILCOM 2012-2012 IEEE Military Communications Conference*. IEEE, 2012.

[15] "Xenon separation VMM"

domains. A service domain may be a network driver domain, a storage driver domain, a security domain, or a service domain that provides some specialized function such as virtual machine introspection.

- **Situation Awareness.** VMM collects security information about each physical host, processing and sharing it.

- **Stub domain**. A simplified domU with limited capabilities and resources. A stub domain runs minimal guest operating systems. It cannot support Linux or Windows. Stub domains typically support a device driver or perform a specialized utility service.

- **Threat.** Harmful outcome, e.g., loss of confidentiality, integrity, availability to one or more domains; threats to control other domains, or entire VMM.

- **Threat model.** Distinguishes the attacker (threat agent), from the threat (harmful outcome), from the misuse case (attacker-product interaction).

- **VCPU**. Virtual CPU of a VM.

- **Virtual memory.** Memory managed within a *guest*; the guest's view of how it is providing virtual memory, using Xen-provided pseudo-physical memory.

- **VM**. Virtual machine.

- **VMM**. *Virtual Machine Monitor* or *hypervisor* is key component of secure hardware sharing. Manages the non-interfering execution of a set of *virtual machines* (VM). The VMM exports *domains* to its interface.