Virtual Machine-Based Trusted Computing Environment and Trusted System Software Design Postprint

Authors: Sun Yuzhong, Haifeng Fang, Ao Ran, Du Lei, Zhang Yanwei, Zhang Kai, Gao Yunwei

Date: 2016-11-02T00:00:00+00:00

Abstract

Data security issues inherent in cloud computing models and reliability problems of traditional system software are of paramount importance for achieving the transformation from traditional computing paradigms to Internet-oriented distributed large-scale concurrent service computing paradigms. This paper focuses on the core software in networked computing platforms, specifically addressing trustworthiness issues, design methodologies, and fundamental architectures of virtual computing environments and system software, and introduces the design scheme of the trusted computing platform Rainbow-T currently under development by our research group.

Full Text

Trusted Computing Environment and Trusted System Software Design Based on Virtual Machines

Yuzhong Sun, Haifeng Fang, Ran Ao, Lei Du, Yanwei Zhang, Kai Zhang, Yunwei Gao

Abstract

Data security issues inherent in cloud computing models and reliability problems in traditional system software are critical for transforming conventional computing paradigms toward Internet-oriented distributed large-scale concurrent service computing. This paper addresses the trustworthiness challenges, design methodologies, and fundamental architectures of core software components in networked computing platforms—specifically, virtual computing environments and system software—and presents the design of Rainbow-T, a trusted computing platform currently under development by our research group.

Keywords: Virtual Machine; Trustworthiness; System Software; Design

1 Introduction

With the rapid advancement of computing and networking technologies, resources such as computation and storage in IT infrastructures are increasingly aggregated through networks. Concurrently, user interaction patterns with computers are shifting from traditional single-desktop computing to large-scale cross-domain service computing centered on the Internet. Consequently, the traditional physical structure of computing is gradually being replaced by networked data centers and similar infrastructure, while computer usage patterns exhibit a service-oriented trend exemplified by cloud computing. In essence, the concept of "The network is the computer" proposed by Sun Microsystems in the 1980s is becoming reality [**?**].

In realizing this vision, extensive research has focused on better sharing of network resources and constructing centralized management views of distributed network resources, leading to various computing models such as utility computing and grid computing. However, practical experience demonstrates that the growth, autonomy, and diversity of network resources impose stringent requirements on networked computing platforms regarding resource management scalability, reliability, and adaptability. Fundamental changes to the core system software architecture of networked computing platforms are necessary, shifting from traditional hierarchical models based on centralized management to distributed flat-structured models based on trusted domains.

Furthermore, networked computing platforms present users with a vast, elastic resource space, resulting in computing environments characterized by openness, dynamism, and difficulty of control [**?**]. These network-based computing environments typically serve large-scale, open user communities with diverse requirements, necessitating support for massively heterogeneous concurrent service computing models. Emerging paradigms such as cloud computing partially satisfy these demands, but inherent conflicts exist between users' established usage habits from relatively closed, static, and controllable environments and the multi-tenant sharing nature of networked computing environments. Addressing this requires user-centric design that provides maximal computing environment isolation while granting users sufficient control—fundamental challenges in building trusted computing environments.

In recent years, system-level virtualization technology (VMM, Virtual Machine Monitor[1]) exemplified by Xen [**?**] has regained prominence and achieved rapid development. System-level virtualization introduces a virtual software layer between hardware and software, transforming underlying physical resources into unified virtual resources. This approach shields applications from hardware platform dynamism, distribution, and heterogeneity while providing each user with

[1]Also known as Hypervisor, translated by some as " 虚拟机监视器"

an independent, isolated personal computing environment. By converting physical resources into virtual ones, virtualization enables resource reuse and sharing, significantly improving utilization in distributed systems and substantially reducing operational costs for enterprise information systems. Virtual computing platforms built upon virtualization technology have become the primary form of enterprise information infrastructure (data centers).

Cloud computing, which has emerged recently, represents a novel computing model primarily targeting virtual computing platforms. Broadly defined, cloud computing presents an elastic, dynamically extensible deployment view of virtualized computing resources (hardware, software, storage, capabilities) provided by service providers based on commercial hardware and software according to utility models. To some extent, cloud computing realizes the Infrastructure as a Service (IaaS) concept. For instance, Amazon' s cloud computing services include Elastic Compute Cloud (EC2) and Simple Storage Service (S3), accessible via the SOAP[2] protocol.

The advent of cloud computing has spurred extensive research in academia and industry, yielding diverse perspectives. UC Berkeley [**?**] views cloud computing as an evolution of utility and grid computing, predicting rapid development driven by major corporations that developers should prioritize. Open-source leader Richard Stallman [**?**] considers cloud computing concepts foolish, arguing that enterprises should never entrust business data to third-party providers. IDC survey results [**?**] identify security, performance, availability, and usability as primary challenges. Customers fear increased attack threats when placing core business information and IT infrastructure outside their firewalls, demand more reliable service quality guarantees from cloud providers, and desire services that integrate tightly with their business environments for easier deployment onto existing infrastructure. These concerns are intimately connected to trust assurance in cloud computing environments, particularly data security issues.

This has sparked intense research interest in virtualized computing platforms and virtual machine-based trust technologies. However, current research in both areas remains exploratory and disconnected, making practical implementation of trusted virtual platforms difficult. This paper focuses on system-level virtualization—the core of virtual machines—to investigate architectural innovations supporting trustworthiness. By integrating multi-core technology, operating system advances, and trust technologies, we aim to build a distributed trusted computing platform for virtual machines that achieves isolation, reliability, and trustworthiness for concurrent virtual machine execution and data, ensuring continuous trusted service delivery despite interference.

Through comprehensive analysis of virtualization technology, trust technologies, and typical cloud computing application models, this paper explores factors, methods, and platform structural design issues related to building trusted virtual computing platforms from the perspective of ensuring distributed system

[2]Simple Object Access Protocol

software reliability and virtual computing environment trustworthiness. It also outlines the design of Rainbow-T, a concrete trusted computing platform under development. Section 2 provides an overview of trusted computing platform concepts and technologies. Sections 3 and 4 analyze trust requirements and solutions for distributed system software and virtual computing environments, respectively. Section 5 introduces Rainbow-T' s overall architecture and subsystems. Finally, Section 6 summarizes key viewpoints, discusses our research progress, and outlines future directions.

2 Basic Concepts and Core Technologies

As mentioned, networked computing introduces new computing environment characteristics such as openness, dynamism, and interactivity, accompanied by new concepts and technologies. From a system designer' s perspective, trust concepts and system-level virtualization technology are crucial.

2.1 Software Trustworthiness

With the rapid development and popularization of the Internet, computing environments have evolved from static, closed, and controllable to open, dynamic, and difficult to control, with application models trending toward collaboration, ubiquity, and service orientation. This evolution emphasizes interaction and cooperation, thereby creating trust establishment challenges.

Research on trustworthiness-related issues is being conducted from various perspectives in academia and industry:

1. **Peking University' s "Basic Theory and Methods for Network-based Complex Software Trustworthiness and Quality of Service"** project focuses on trust assurance for Web application software quality of service in Internet environments. Based on Internetware, it aims to establish a novel quality assurance system "centered on usage quality and grounded in system quality" to drive new software methodologies and technical systems.
2. **EMC China Research Center' s "Daoli" project** [?] is a trusted grid initiative based on Xen[3] and Trusted Platform Module (TPM). It aims to enhance grid platform trustworthiness and security through behavior verification by introducing assurance mechanisms at both operating system and middleware layers.
3. **UC Berkeley' s TRUST project** [?] focuses on cybersecurity issues, establishing new scientific and technological foundations for designing, building, and operating trusted information systems.
4. **The Trusted Computing Group (TCG)** [?], initiated by IBM and other companies, addresses client-side untrustworthiness issues. Centered

[3]An open-source virtual machine monitor developed by Cambridge University

on platform integrity and through TPM, it enables trust verification during server interactions to establish trusted communication.

5. **Europe's OTC project** [**?**] combines TCG's trusted computing hardware facilities with virtualization' s isolation mechanisms to propose a secure and trusted system construction and development scheme based on open-source software.

These research efforts primarily focus on trust issues during interaction processes, covering trustworthy assurance for usage patterns, trusted deployment of software in computing environments, and trusted interaction in open computing environments. Additionally, the National Natural Science Foundation of China' s 2007 "Basic Research on Trusted Software [**?**]" major research program proposed shifting from the traditional "correctness-oriented software theory + engineering" model to a software trustworthiness measurement-based approach that comprehensively considers trust issues throughout the entire software engineering lifecycle and runtime support environment.

Researchers have proposed numerous trust-related concepts from different perspectives:

- **High Confidence**: From a software quality perspective, emphasizing that software system behavior results can be well understood and predicted, requiring verifiable software that meets basic correctness proof requirements. Primarily examines attributes including correctness, predictability, safety, reliability, and survivability.
- **Dependability**: Originating from early reliability and fault-tolerance concepts, this perspective focuses on software structure and runtime assurance, requiring tolerance for existing defects and external erroneous inputs while maintaining service quality metrics. Primarily examines availability, reliability, safety, integrity, and maintainability.
- **Trustworthiness**: Going beyond dependability, this behavioral perspective requires systems to provide evidence regarding expected execution, including execution results, environment, behavior logs, etc., enabling trustworthiness assessment. Primarily examines security, privacy, reliability, and business integrity.
- **Trusted Computing**: From an interaction perspective, focusing on trust management, requiring trust establishment among interaction participants and within collaborative domains formed according to specific objectives.

These concepts reflect preliminary explorations of trust, essentially combining traditional concepts like correctness, reliability, security, integrity, availability, predictability, and controllability from different angles, without yet achieving unified understanding.

We believe software trustworthiness should be addressed comprehensively from a software engineering perspective, with different measures and trust attributes

at each stage:

- **Design Phase**: Through modeling and formal methods, software quality can be verified to assess program quality. For system software design, besides quality assurance, trust assurance services for the runtime environment must be considered. This phase focuses on reliability and scalability as system software trust attributes.
- **Development Phase**: Quality assurance is achieved through reliable software tools and effective development process management.
- **Runtime Phase**: For trust requirements difficult to guarantee during design and development (e.g., performance, reliability) and security requirements arising from anomalies in open, dynamic environments, corresponding system software trust assurance measures must be deployed. This phase focuses on safety/security, privacy, and integrity as runtime support environment trust attributes.

2.2 System-Level Virtualization Technology

System-level virtualization emerged in the 1960s when mainframe computers were extremely expensive. System designers sought to enable large numbers of users to share access to mainframes, reducing procurement costs while improving server utilization. Additionally, different users desired different operating systems. These needs gave rise to system-level virtualization. However, with the PC era, demand for concurrent shared access to large servers diminished, causing the technology to fade from mainstream use. The recent emergence of networked computing has aggregated various physical resources through networks, forming a massive networked supercomputer where large-scale multi-user concurrent access has again become a key design issue, leading to renewed emphasis and rapid development of system-level virtualization on this new physical platform.

System-level virtualization is a relatively low-level technology that essentially uses software or hardware to generate more virtual nodes than physical resources, fully compatible with actual nodes for upper-layer application software. It provides instruction set interface consistency and physical server virtualization. Its basic structure is shown in Figure 1 [Figure 1: see original paper].

In 1974, Popek and Goldberg [**?**] established basic requirements for CPUs supporting system-level virtualization, though most existing CPUs, including Intel IA-32 architecture, did not provide virtualization support. Consequently, system-level virtualization must be implemented through software. Based on whether guest operating system modifications are required, system-level virtualization can be divided into full virtualization and para-virtualization systems.

Full virtualization, exemplified by VMware, allows unmodified guest operating systems to run directly. Guest OS code execution typically combines direct execution with dynamic binary translation, translating non-virtualizable IA-32

instructions to compensate for processor limitations. The primary disadvantage of full virtualization is relatively low efficiency.

Para-virtualization, exemplified by Xen, requires replacing non-virtualizable IA-32 instructions and some privileged instructions with alternative instruction sequences, necessitating guest OS modifications. Para-virtualization achieves higher efficiency than full virtualization, garnering widespread attention and rapid development. Cloud platforms from Amazon and IBM use Xen as their core virtualization technology. Xen' s structure is shown in Figure 2 [Figure 2: see original paper].

Virtualization technology, particularly system-level virtualization, has gained prominence in cloud computing platforms due to its core principles:

Resource Reuse: As server configurations and performance improve and enterprise IT infrastructure expands, resource utilization has decreased rather than increased. The industry recognized the need for technologies to fully leverage existing infrastructure. Representative work includes: - **Disco** [**?**] manages multiprocessor systems based on ccNUMA[4] architecture, supporting multiple upper-layer operating systems through virtualization. Disco globally manages physical resources like memory and processors, enabling memory migration among virtual machines, dynamic processor binding and scheduling for load balancing, while shielding upper OSes from memory distribution sensitivity. - **VMware**, the full virtualization representative, enables unmodified OS execution, partitions resources like memory at the virtual machine level, and introduces overcommit techniques for processors and memory to maximize virtual machine density. - **Xen**, the para-virtualization representative, naturally separates virtual computing environments from storage and other resources, enabling separate management of memory, processor, and communication resource reuse to form different resource management pools.

Environment Isolation: Early mainframes using multi-terminal shared access required time-sharing resource multiplexing and security isolation among users. Multics[5] introduced security protection mechanisms, most notably protection rings and access control lists, laying groundwork for system-level virtualization research, particularly for designing security kernels—trusted computing bases (TCB). VAX was the earliest machine providing protection rings, offering a platform for creating system-level virtualization. VAX VMM [**?**] utilized a VAX ring to implement the virtualization layer and hierarchically constructed a security kernel.

Virtualization achieves runtime isolation for different applications, enabling utilization of isolation mechanisms for various objectives: - **Providing different virtual computing environments based on application trust requirements**: Stanford' s Terra [**?**] divides virtual machines into "open box" and

[4]Cache-Coherent Non-uniform Memory Access

[5]MULTiplexed Information and Computing System, developed by Bell Labs, MIT, and GE in 1964

"closed box" environments, with the latter monitored and verified by trusted computing components. Proxos [**?**] from the University of Toronto allows applications to specify OS trustworthiness requirements, providing restricted virtual machines that forward sensitive system calls to trusted VMs. - **Classifying virtual machines by functional roles**: Critical services can be isolated through virtualization to ensure reliable component operation. In Xen, the domain builder (DomB) [**?**] encapsulates VM creation components into dedicated VMs, while DomT isolates TPM management and verification components. For compute-intensive applications, customized StubDom [**?**] can encapsulate them in specialized VMs to improve efficiency.

This section provides an overview of trust concepts and virtualization technology, establishing a fundamental framework for trusted computing environment and system software design from both requirements and solutions perspectives.

3 Trusted System Software

Networked computing platforms are essentially constructed through large-scale aggregation of computing nodes, network-based storage nodes, and particularly Internet-based cross-domain resources—an evolution of cluster architectures. Corresponding system software must therefore be distributed. Traditional distributed system software primarily represents network-oriented extensions of single-machine operating systems, focusing on resource integration and performance optimization with monolithic kernel architectures (some employing microkernel structures). However, their orientation toward static physical resource global management severely limits scalability. Internet-oriented networked computing must satisfy large-scale multi-user concurrent access, demanding greater flexibility for on-demand resource scaling and more reliable quality assurance—emphasizing trust attributes like scalability and high reliability.

Current system software, particularly operating systems, exhibits several deficiencies in Internet application environments: - **Monolithic kernel structure**: Historically pursuing performance optimization, most OSes employ monolithic kernels whose single points of failure can cause complete system crashes, making fault isolation extremely difficult. - **Excessive software layers**: For design and maintenance simplicity, OSes often adopt top-down hierarchical designs. As system functionality increases, layers multiply, lengthening execution paths and significantly reducing efficiency. - **Generic interface abstraction**: Monolithic and hierarchical designs require extensive generic interfaces to support diverse applications, creating complexity similar to Complex Instruction Set Computing (CISC) processors and causing software deployment redundancy. - **Numerous software defects**: As system software complexity increases, scale becomes enormous—the Linux kernel exceeds 2.5 million lines of code. Software reliability research indicates approximately 6+ defects per thousand lines, suggesting around 15,000 defects in Linux. - **Complex internal function call relationships and hundreds of system call interfaces create numerous potential fault points and error input vulnerabilities, posing**

significant security risks. - Implementation in traditional C language easily introduces defects like buffer overflows.

These deficiencies severely reduce system reliability and trustworthiness, requiring targeted improvements. Current academic proposals include: - **Microkernel architecture**: Microkernels offer natural fault isolation and environment customization, with current distributed architectures and multi-core platforms providing a practical foundation. Existing virtualization systems essentially employ microkernel structures, offering fundamental solutions to monolithic kernel problems. - **Reduced system-level virtualization scale**: The security and reliability of the virtualization core are critical, requiring continuous removal of non-core functions to minimize code size and form a verifiable trusted computing base core. To simplify the TCB while maintaining functionality, less trusted components can be connected through trusted encapsulation with controlled communication interfaces, effectively reducing TCB size. - **Driver isolation through protective wrappers**: Drivers represent the largest fault source in OS kernels. Nooks[6] [**?**] adds protective wrappers to each driver, monitoring all driver-kernel interactions to effectively isolate driver-induced faults. Similarly, Xen' s Driver Domain uses virtual machines to encapsulate drivers. - **Simplified interaction interfaces**: Reanalyzing interfaces between critical components and specializing them for particular applications prevents other software from accessing these key interfaces. Xen applies this concept by exposing VM creation interfaces only to DomB. KVM[7] [**?**] defines clear interaction interface specifications, constraining and standardizing interactions between core and untrusted components to provide code-level trust assurance. - **Type-safe and verifiable languages**: Using such languages improves system software quality, particularly reducing buffer overflows. Microsoft' s Singularity[8] system uses Sing# (based on C# with message-passing primitives), strictly limiting system and user process behavior with compiler-enforced prohibition of cross-process data access. - **Enhanced security monitoring modules**: These ensure runtime trustworthiness, such as IBM' s OS-independent sHype/xsm [**?**] security architecture introducing Mandatory Access Control (MAC) in Xen, and SUNY Stony Brook' s FVM [**?**] process behavior monitoring framework for Windows.

Overall, improving system software trustworthiness can address software defect analysis, fault isolation, and language-based protection. However, these methods have limitations, only partially improving trust attributes. As information system infrastructure becomes networked, flattened, and distributed, corresponding system software—particularly operating systems—should evolve from vertical hierarchical monolithic structures to horizontal flat microkernel architectures. Only structural transformation can fundamentally improve system software trustworthiness.

[6]A subsystem for enhancing OS reliability developed by University of Washington PhD student M. Swift in 2005

[7]A microkernel virtual machine building security with clear interfaces

[8]An OS developed by Microsoft Research for reliability studies

Such transformation must maintain backward compatibility with existing applications, remaining transparent to them. Virtualization technology ensures this, but simple VM encapsulation (which would reinforce traditional hierarchical architecture problems) must be avoided. Instead, design should target trust objectives like scalability and high reliability through granular, stateless, and dynamic structural design.

Trusted system software for networked computing platforms should provide: - **Availability**: Resource and service quality assurance - **Scalability**: On-demand dynamic deployment capabilities for computing environments to adapt to changing physical resources and demands - **Reliability**: Guaranteed normal operation of critical components - **Integrity**: Protection of platform and data information completeness - **Security**: Attack resistance for external interactions, ensuring normal operation under interference

To address these trust requirements, our VM-based system software design employs these measures:

1. **Leveraging Xen' s para-virtualization and compute-storage separation**: Utilizing Distributed Virtual Machine Monitor (DVMM) global resource management capabilities to ensure availability through resource on-demand mobility.
2. **Providing customized lightweight virtual computing environments**: Tailored to application characteristics with flexible composition and binding to virtual resources for rapid deployment.
3. **Utilizing VM isolation for reliability**: Through kernel splitting and customization, providing specialized VMs for critical components (e.g., device management, monitoring) to improve reliability.
4. **Establishing flat system structure**: Applications can directly bind to critical components in a VM-based network composition structure. A trust chain built on trusted computing modules ensures overall structural integrity.
5. **Domain-based VM partitioning**: For multi-service operation, providing transparent behavioral security monitoring mechanisms for interactions between domains, VMs, and compute-storage components to effectively resist external interference.

4.1 Virtual Computing Environment Trust Requirements

Computing paradigms have cycled from early mainframe multi-terminal time-sharing systems to PC popularity (centralized to distributed resources), then through network-based grid systems to today' s Internet-based cloud computing (recentralized resources). Clouds resemble early mainframes with networked computing environments as their foundation, typically in data center form. Unlike early time-sharing systems, cloud computing primarily employs space multi-

plexing, focusing on flexible resource composition and management. Clouds can be categorized as compute clouds, storage clouds, etc., based on resource types. Resource composition requires negotiation between users and providers, with providers publishing resource services through market mechanisms and users (individual or enterprise) procuring and customizing resources to establish suitable computing architectures ranging from simple single-machine services to complex grid systems. Cloud computing' s on-demand resource capability also enables effective user resource extension—enterprises can request supplementary resources from cloud providers when their own are insufficient. Stanford University' s Collective system [**?**] developed virtual appliances to reduce user system management burden and enhance application/data trustworthiness. By embedding resource extension modules in resource-constrained front-end devices like laptops, users can dynamically acquire backend resources through cloud computing, effectively combining cloud resource utilization with flexible front-end access.

Current cloud computing service forms include: - **Software-as-a-Service (SaaS)**: Providing online software services via the Internet - **Platform-as-a-Service (PaaS)**: Providing development platforms with APIs for cloud-oriented software development, such as Google App Engine' s Python[9] framework for accessing databases, image processing, URLs, and other web resources to build Web applications - **Infrastructure-as-a-Service (IaaS)**: Providing flexible composition and usage of computing and storage resources, exemplified by Amazon' s cloud computing services

UC Berkeley identifies three major issues for widespread cloud computing adoption [**?**]: 1. **Service availability**: Requiring powerful extensibility, such as deploying computing services across multiple cloud providers for dynamic capacity extension and improved attack resistance. 2. **Data controllability**: Preventing data management from being constrained by specific service providers and ensuring storage format compatibility across services for on-demand data location adjustment. 3. **Data protection and auditing**: Providing encrypted data transmission, access logging, and regulatory compliance for data storage locations.

These issues converge on fundamental requirements for improving cloud computing trustworthiness, particularly data security. Current technical measures include:

1. **Dynamic cross-domain deployment for compute clouds**: Such as Linux Virtual Server (LVS) and load balancing-based dynamic service location adjustment. The SnowFlock prototype [**?**] from the University of Toronto and Carnegie Mellon enables dynamic compute environment distribution across physical nodes for parallel computing tasks. Columbia University' s Remote Fork mechanism [**?**] provides OS-level cross-node process distribution for concurrent execution across multiple virtual com-

[9]An open-source scripting language

puting environments.

2. **Flexible composition and binding for storage clouds**: Virtual appliances[10] [**?**] simplify application/data deployment through image encapsulation, partitioned into system, data, and temporary storage types for flexible access and data isolation. Parallax [**?**] from the University of British Columbia provides template-based image management with snapshot-based protection and redundancy sharing as a block-level storage pool management system.

3. **Transparent monitoring and management for control clouds**: Using trusted roots and TPMs to build trust chains from system boot through virtual computing environment runtime for trust measurement. IBM' s sHype introduces monitoring mechanisms for virtual computing environment and storage resource interactions in Xen.

Given user concerns about data security, the popularity of pervasive computing, and increasing emphasis on cloud computing openness and interoperability, compute-storage separation has become essential. Providing IaaS based on compute-storage separation is critical for strengthening user control over computing environments and storage resources, thereby improving trustworthiness.

Based on this analysis, we believe trusted computing environments for networked computing built on VM-based distributed system software should be virtual computing platforms founded on compute-storage separation, enabling on-demand flexible composition of computing environments and storage resources to provide "Infrastructure as a Service."

4.2 Virtual Computing Platform Trust Assurance Technologies

Compute-storage separation-based virtual computing platforms require system-level virtualization, particularly Xen' s para-virtualized device management model that achieves virtual computing and storage separation. This model enables separate consideration of compute environment and storage resource security and trustworthiness, with their trusted binding being fundamental to overall platform trustworthiness.

Virtual computing environments built on virtualization exhibit dynamic characteristics [**?**]: - **Compute environment dynamism**: (1) Hardware facilities may change at any time due to VM hardware abstraction; (2) Software deployed in VMs can be dynamically migrated; (3) VMs have rich lifecycle states with on-demand status adjustment capabilities. - **Network service dynamism**: (1) VM MAC addresses are random at creation and change with VM state,

[10] An on-demand resource invocation mechanism where applications encapsulated in VMs can rapidly create runtime environment copies on distributed physical hosts using fork-like functions, releasing all resources immediately upon completion

causing broadcast/multicast service failures; (2) VM state dynamism and mobility break long-term one-to-one relationships between IP addresses and VMs, rendering DNS mostly ineffective. - **Storage service dynamism**: Storage primarily serves as images, with local storage often used as temporary buffers and persistent storage typically existing as remote resource pools.

This dynamism breaks traditional security mechanisms'static assumptions about protected platform states (machine count, configuration, location), causing security mechanisms to be stripped from VMs. Current trust assurance technologies for virtual platforms focus on:

- **Code integrity**: Ensuring running software is unmodified and not maliciously altered during execution to guarantee predictable behavior
- **Data secrecy**: Ensuring memory and stored files are not arbitrarily accessed or exposed
- **Monitoring transparency**: Providing transparent VM monitoring measures in untrusted computing environments to improve monitor attack resistance
- **Verifiable interaction**: Requiring interaction parties to verify conditions and establish trust before allowing interaction

Specific research includes:

1. **Application-oriented runtime**: Process-level trusted isolation through effective management of process address spaces (page tables) to prevent unauthorized private resource access. Representative work includes:
    - **SP3** [**?**] from the University of Michigan assumes GuestOS is untrusted, bypassing the OS to use system-level VMs for application data protection. Applications can request page encryption directly from the VM, preventing the OS from accessing plaintext even when mapping pages to other processes.
    - **Overshadow** [**?**] from VMware, Stanford, Princeton, and MIT provides similar SP3 mechanisms, intercepting process-kernel interactions to protect trusted process CPU contexts and memory page tables.
    - **SIP** in Microsoft' s Singularity system uses type-safe language design requiring processes to communicate only through verifiable message channels without data sharing, with code labeled and isolated through software verification.
    - **Loki** [**?**] from MIT and Stanford uses tagged memory architecture in processors to support tag-based memory management for OS information flow monitoring.
2. **Subject-object trusted interaction**: Monitoring process resource access behavior. Representative work includes:
    - **XenAccess** [**?**] from Georgia Tech observes guest OSes through cross-domain memory mapping in Xen control interfaces.
    - **Xenprobe** [**?**] from Japan' s National Institute of Advanced Industrial Science and Technology dynamically inserts hook functions in

guest OSes for process behavior capture.

- **SecureBus** [**?**] from George Mason University implements an information channel between processes based on User-Mode Linux, introducing reference monitors[11] to control inter-process interactions.
- **Antfarm** [**?**] from the University of Wisconsin-Madison detects suspicious processes like Trojans by observing hardware behaviors related to process address spaces (CR3 changes, TLB flushes) and comparing with OS-provided process information without requiring guest OS details.
- **Vmwatcher** [**?**] from George Mason University and Purdue provides out-of-band VM monitoring using OS key data structure definitions as templates to infer actual runtime information (e.g., process count) for antivirus software.
- **Lares** [**?**] from Georgia Tech also provides out-of-band monitoring by dynamically inserting modules with hook functions for critical monitoring paths, analyzing self-protection and VM memory protection requirements to prevent malicious tampering.

3. **Platform configuration integrity measurement**: Representative work includes:
   - **vTPM** [**?**] from IBM introduces a virtual device in VMs for accessing TPMs to measure loaded system and application software integrity.
   - **Copilot** [**?**] from the University of Maryland uses a coprocessor board with DMA to periodically obtain kernel memory critical sections and detect tampering.
   - **MIF** [**?**] from IBM and the University of Virginia is a new image format that separates file-block mapping from traditional images for rapid internal structure acquisition and storage-based monitoring.

In summary, current approaches for building trusted runtime support environments are based on trust measurement technology, introducing trust enhancement measures around computing environment security and storage resource trustworthiness.

5 Rainbow-T Virtual Computing Platform

Our research group' s Rainbow platform provides Amazon-like computing environments built on LAN infrastructure, offering resource flow-based management for full utilization. The underlying distributed system-level VM provides global resource management pools for computing, storage, and memory, with Xen enabling on-demand dynamic deployment of virtual application computing environments. Compute and storage separation enables flexible, on-demand composition through resource flow, allowing dynamic application deployment that triggers resource flow and improves utilization through service integration. The basic framework is shown in Figure 3 [Figure 3: see original paper].

[11] Access control programs that monitor and control subject access to data and code objects

Based on our understanding of trusted computing environments and centered on splitting, customization, reconstruction, and isolation enhancement, we have introduced the following system functions and subsystems from trusted system software and trusted computing environment perspectives:

1. **Scalable computing environment for applications**: A cross-domain distributed execution environment for multi-threaded shared-memory applications providing transparent runtime support for large-scale concurrent multi-threaded applications like traditional Web servers, enabling arbitrary deployment across multiple physical or virtual nodes within a trusted domain.

2. **Kernel trustworthy reconstruction through splitting and customization**: From an application service perspective, reducing unnecessary software layers to provide customized lightweight virtual computing environments for easy deployment to other nodes, such as application-specific customized runtime environments. From a device management perspective, addressing kernel device management unreliability through VM isolation to provide a dedicated device driver domain for reliable resource management.

3. **Verification-based service trusted isolation**: Addressing trusted domain composition for VM-based application computing environments through VM partitioning by application service, using TPM-based VM environment measurement as the primary trust metric, providing dynamic trusted domain admission mechanisms, and introducing heterogeneous service domain-based trusted network isolation for secure concurrent service domain coexistence.

4. **Trusted interaction between computing environments and storage resources**: Compute-storage separation requires separate trust assurance and trusted interaction considerations for computing and storage, while granting users reasonable control capabilities. Remote binding technology enables user requirement-oriented compute-storage composition, with process behavior-based application isolation monitoring maximizing user control over data resource access.

These key technologies constitute the main content of the Rainbow-T trusted virtual computing platform, which our research group is currently developing in depth.

6 Conclusion

This paper introduced the origins of trust issues and related research, arguing that trusted computing environment and system software design should adopt a software engineering perspective focusing on dependability and trustworthiness attributes. Through analysis of system-level virtualization research, we identified virtualization' s core principles, noting that its isolation concept

provides technical assurance for trusted system software design. Using virtualization isolation for system software structural reconstruction and leveraging compute-storage separation to provide data security-oriented virtual computing environment trust assurance mechanisms represent the primary approach for achieving system software and computing environment trustworthiness. Based on this approach, we presented the design philosophy and key technologies of the Rainbow-T trusted virtual computing platform.

References

[1] Xicheng Lu, Huaimin Wang, Ji Wang. iVCE: Concepts and Architecture of Virtual Computing Environment. Science in China (Series E), 2006, 36(10):1081-1099 [2] Jian Lv, Xiaoxing Ma, Xianping Tao, et al. Research and Development of Internetware. Science in China (Series E), 2006, 36(10):1037-1080 [3] Paul Barham, Boris Dragovic, Keir Fraser, et al. Xen and the Art of Virtualization. In: Proc. of the 19th ACM Symp. on Operating Systems Principles(SOSP). 2003: 164 - 177 [4] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia. Above the Clouds: A Berkeley View of Cloud. Technical Report No. UCB/EECS-2009-28, 2009 [5] Richard Stallman: Cloud Computing a Trap. http://www.linux-magazine.com/Online/News/Richard-Stallman-Cloud-Computing-a-Trap [6] Exclusive: The Bill Gates Exit Interview. http://www.pcmag.com/article2/0,2817,2321132,00.asp [7] http://www.grid.org.il/Uploads/dbsAttachedFiles/IDC_{{{Cloud}}*{{{Computing}}}*{{{IGT}}}_{{final}}}.pp [8] http://www.daoliproject.org/ [9] http://www.truststc.org/ [10] http://www.trustedcomputinggroup.org/ [11] http://www.opentc.net/ [12] Ke Liu, Zhiguang Shan, Ji Wang, Jifeng He, Zhaotian Zhang, Yuwen Qin. Overview of the Major Research Program "Basic Research on Trusted Software" . Bulletin of National Natural Science Foundation of China, 2008, Issue 3, 145-151 [13] Jim Smith, Ravi Nair. Virtual Machine: versatile platform for systems and processes. Morgan Kaufmann, 2005 [14] Gerald J. Popek, Robert P. Goldberg. Formal requirements for virtualizable third generation architectures. Communications of the ACM, 1974, Vol. 17(7): 412 - 421 [15] Edouard Bugnion, Scott Devine, Kinshuk Govil, et al. Disco: Running commodity operating systems on scalable multiprocessors. ACM Transaction on Computer Systems (TOCS), 1997, Vol. 15(4): 412-447 [16] Paul A. Karger, Roger R. Schell. Thirty Years Later: Lessons from the Multics Security Evaluation. At the 18th Annual Computer Security Applications Conference (ACSAC) on December 9-13, 2002 in Las Vegas, Nevada, 2002: 119 [17] Paul A. Karger, Mary Ellen Zurko, Douglas W. Bonin, et al. A VMM Security Kernel for the VAX Architecture. In: IEEE Symposium on Security and Privacy (S&P), 1990: 2-19 [18] Tal Garfinkel, Ben Pfaff, Jim Chow, et al. Terra: A Virtual Machine-Based Platform for Trusted Computing. In: Pro. of the 19th ACM Symp. on Operating Systems Principles(SOSP), 2003: 193 - 206 [19] Richard Ta-Min, Lionel Litty, David Lie. Splitting Interfaces: Making Trust Between Applications and Operating

Systems Configurable, In: Proc. of the 7th Symp. on Operating Systems Design and Implementation(OSDI), 2006: 279 - 292 [20] Derek Gordon Murray, Grzegorz Milos, Steven Hand. Improving Xen Security through Disaggregation. In: Proc. of the 4th ACM SIGPLAN/SIGOPS international conference on Virtual Execution Environments (VEE), 2008: 151-160 [21] Samuel Thibault, Tim Deegan. Improving performance by embedding HPC applications in lightweight Xen domains. In: Proc. of the 2nd workshop on System-level virtualization for high performance computing (HPCVirt), 2008: 9-15 [22] Andrew S. Tanenbaum, Jorrit N. Herder, Herbert Bos. Can We Make Operating Systems reliable and Secure? IEEE Computer, 2006, Vol. 39(5): 44 - 51 [23] Xiaoqi Lu, Scott F. Smith. A Microkernel Virtual Machine Building Security with Clear Interfaces. In: Proc. of the 2006 workshop on Programming languages and analysis for security, 2006: 47 - 56 [24] R. Sailer, E. Valdez, T. Jaeger, et al. sHype: Secure Hypervisor Approach to Trusted Virtualized Systems. Techn. Rep. RC23511, Feb. 2005. IBM Research Division [25] Yang Yu, Hariharan Kolam govindarajan, Lap-Chung Lam, Tzi-cker Chiueh. Applications of a Feather-weight Virtual Machine. In: Proc. of the 4th ACM SIGPLAN/SIGOPS international conference on Virtual Execution Environments (VEE), 2008: 171-180 [26] Ramesh Chandra, Nickolai Zeldovich, Constantine Sapuntzakis, Monica S. Lam. The collective: a cache-based system management architecture. In: Proc. of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI), 2005, Vol.2: 259-272 [27] H. Andrés Lagar-Cavilla, Joseph Whitney, Adin Scannell, et al. Impromptu Clusters for Near-Interactive Cloud-Based Services. Technical Report CSRG-TR578, Department of Computer Science, University of Toronto, June 2008 [28] Jonathan M. Smith, John Ioannidis. Notes on the Implementation of a Remote Fork Mechanism. Technical Report #CUCS-275-87, Columbia University Computer Science Department (1987) [29] David Lutterkort, Mark McLoughlin. Manageable Virtual Appliances. At Linux Symposium 2007, Ottawa, Canada [30] Dutch T. Meyer, Gitika Aggarwal, Brendan Cully, et al. Parallax: virtual disks for virtual machines. In Proc. of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys) 2008: 41-54 [31] Tal Garfinkel, Mendel Rosenblum. When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments. In: Proc. of the 10th conference on Hot Topics in Operating Systems (HotOS) 2005: 20-20 [32] Jisoo Yang, Kang Shin. Using Hypervisor to Provide Application Data Secrecy on a Per-Page Basis. In: Proc. of the 4th ACM SIGPLAN/SIGOPS international conference on Virtual Execution Environments (VEE), 2008: 71-80 [33] Xiaoxin Chen, Tal Garfinkel, E. Christopher Lewis, et al. Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems. In Proc. of the 13th international conference on Architectural support for programming languages and operating systems (ASPLOS), 2008: 2-13

Note: Figure translations are in progress. See original paper for figures.