Enabling Secure Multitenancy in Cloud Computing: Challenges and Approaches

Takeshi Takahashi*, Gregory Blanc†, Youki Kadobayashi†, Doudou Fall†, Hiroaki Hazeyama†, Shin'ichiro Matsuo*
* National Institute of Information and Communications Technology, Tokyo, Japan
Email: takeshi_takahashi@nict.go.jp
† Nara Institute of Science and Technology, Nara, Japan

***Abstract*—Cloud computing provides a multitenant feature that enables an IT asset to host multiple tenants, improving its utilization rate. The feature provides economic benefits to both users and service providers since it reduces the management cost and thus lowers the subscription price. Many users are, however, reluctant to subscribe to cloud computing services due to security concerns. To advance deployment of cloud computing, techniques enabling secure multitenancy, especially resource isolation techniques, need to be advanced further. Difficulty lies in the fact that the techniques range and cross various technical domains, and it is difficult to get the big picture. To cope with that, this paper introduces technical layers and categories, with which it identifies and structures technical issues on enabling multitenancy by conducting a survey. Based on the survey result, this paper discusses technical maturity of multitenant cloud computing from the standpoint of security and the needs for developing both technical and operational security toward the development and wide deployment of multitenant cloud computing.**

***Index Terms*—Multitenancy, cloud computing, security, resource isolation**

I. Introduction

The development of cyber societies and online transactions imposes continuously expanding IT budgets on organizations. To handle this, organizations are redesigning their procurement and management strategies for IT infrastructure. Cloud computing services become their candidate solutions since they provide economic benefits; they reduce hardware and software expenses while canceling out related maintenance and upgrade costs. They offer on-demand, flexible access to appropriate amounts of computation, memory, and storage resources. The advantage is brought by their multitenant feature, which enables an IT asset to host multiple tenants.

Though cloud computing has quickly gained popularity, many organizations are still reluctant to use the services in consideration of potential security threats to their data. Multitenant feature must ensure that each tenant's data is kept confidential so that only authorized tenants can access it. Thus resource isolation techniques are one focus of multitenancy's security issues. To advance the development and deployment of cloud computing, security issues and approaches for the techniques need to be clarified. Difficulty lies in the fact that these techniques range and cross various technical domains including cryptography, virtualization, and programming.

To facilitate and advance the development of cloud computing, this paper introduces technical layers and categories, with which it identifies and structures security issues of multitenant techniques by conducting a survey. The security issues and techniques addressed in this survey are not limited to a specific technical domain but span assorted such domains since multitenancy is enabled by various techniques, but this survey focuses on issues related to resource isolation and depicts one snapshot of such security techniques to maintain the interests of readers. Based on the result, this paper discusses the technical maturity of multitenant cloud computing from the standpoint of security and discusses the needs for developing both technical and operational security toward wider deployment of multitenant cloud computing.

The rest of the paper is organized as follows: Section II introduces the technical layers and classifications, Section III describes the survey methodology, Sections from IV to VIII address security issues for each of the technical layers, Section IX discusses technical maturity and the needs for advancing security operations, and Section X concludes the paper.

II. Technical Layers and Classifications

This section introduces technical layers and categories that are useful to get a big picture of multitenancy techniques. Cloud computing services are often classified into three varieties: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS); the latter giving less control to end-users and the former requiring that they ensure their own security. Though this classification is useful for discussing business issues and the degree of control from a customer standpoint, another classification is needed to discuss technical issues.

This paper defines five technical layers and their categories, as in Table I. HW&SW primitive layer techniques are fundamental to the other layers' techniques, and are comprised of programming code and hardware processing techniques. Hypervisor layer techniques isolate tenants by providing separate virtual machines (VMs). The isolation is securely enabled if both VMM and VM are functioning properly. OS-layer techniques isolate users by isolating their processes, thus the customizability of the isolation for each user is bound to that of processes, and the isolation is securely enabled if privilege is separated and kernel is functioning properly. Application layer techniques isolate logical resources divided by a database for each user so that the isolation is not bound to process and

TABLE I
TECHNICAL LAYERS AND CATEGORIES

Layers	Categories
Web	Server-side security
	Client-side validation
Application	Secure data storing
	User data isolation
	Authentication and authorization
Operating System	Privilege separation
	Kernel integrity
Hypervisor	VMM security
	VM security
HW&SW Primitive	Programming code
	Hardware processing

is fully customizable in the application layer. Data needs to be securely stored and isolated between users, and access to that needs to be authenticated and authorized to enable secure isolation. Web layer techniques distinguish users and accept their accesses to multitenant cloud computing services over networks. Any incident in this layer may cause malfunction in the other layers. To avoid that, server-side application needs to be secured and client-side application needs to be validated.

III. SURVEY METHODOLOGY

This paper surveys literature addressing security issues and techniques of multitenant cloud computing. The security issues and techniques addressed in this survey are not limited to a specific technical domain but span assorted such domains since multitenancy is enabled by various techniques. We mainly surveyed journals, papers, and technical notes published by ACM, Black Hat, IEEE, Usenix, and the other notable conferences from 2002 to 2011, and summarized prominent ones. Though this paper identifies emerging or renewed major security issues related to multitenant cloud computing, it does not provide an exhaustive list of such issues since there exist a tremendous amount of techniques supporting multitenancy, and that list is of very little interest to the research community. The survey describes one snapshot of security issues of multitenant cloud computing, focusing on resource isolation.

IV. HW&SW PRIMITIVE LAYER SECURITY

Multitenancy service developers usually assume that underlying hardware and software primitive tools are reliable and secure, but that is not always the case.

A. Programming Code

A software is a set of programming codes, which is carefully developed by software developers to avoid software bugs causing security flaws. Some programming codes, however, ultimately contain security risks by their nature. The risks and countermeasures are elaborated below.

1) Managed Runtime Security: While .NET CLR and Java VM have long been considered trustworthy software, in reality their security is rapidly becoming mediocre.

Two major managed code platforms, .NET and Java, could be abused by planting malicious code inside the Framework and managed code runtime [1], [2]. In case of .NET MSIL, the abuse is taken place by locating a target DLL in the Global Assembly Cache (GAC), analyzing it, modifying its MSIL code, recompiling the code to a new DLL, then deploying the new DLL while overwriting the original. With the above technique, an attack can plant malicious code inside the framework. Such code is not easily detected: code reviews will not detect backdoors installed inside the Framework since the payload is not in the code itself, but rather it is inside the Framework implementation. The techniques is a post exploitation type attack, and the similar steps are applicable to JAVA's JVM.

On the other hand, JavaSnoop [3] allows introspection of code executed under Java VM, e.g. interception of method calls, alteration of parameters and return values, and code insertion, without access to the source code. Traditionally, Java programs seem to be unrelated to this dynamic introspection problems in the Web application world. The arrival of JavaSnoop will refute the long-standing implicit assumption that Java code and data flows cannot be tweaked on the fly.

2) Sandboxing: Sandboxing is an effective means of eliminating side effects of executing untrusted binaries in a hosted environment. Sandboxing could be enabled by kernel extension or by modifying guest codes. Different from these, Vx32 [4] is a user-level sandboxing that does not require above modifications. It implements code sandboxing through a combination of dynamic binary translation and x86 segmented address space. Their translation mainly instruments branching instructions, keeping the rest of the unprivileged x86 instructions intact. While Vx32 provides a good isolation feature in terms of memory, data contamination through I/O may occur. Data and network layer isolation can be implemented by system call interposition and type enforcement techniques.

B. Hardware Processing

Multitenancy service developers usually assume that isolation works as intended, both at the software and hardware layers. Ultimately the isolation feature may fail on certain occasions due to the innovative nature of software.

The presence of side channels is identified in the shared memory hierarchy of a Pentium 4 with hyper-threading features [5]. Both the Level-1 and Level-2 caches of the Pentium 4, with the hyper-threading feature turned on, can lead to information leakage from one process to another, potentially hostile, process. For instance, OpenSSL keys could be theft.

An examination on instruction sets of Intel Pentium architecture is reported [6], It identified several sets of sensitive (e.g., unprivileged but hazardous) instructions and concluded that current virtual machine monitors (VMMs) for Intel architecture should not be used to enforce critical security policies.

V. Hypervisor-Layer Security

Hypervisor-layer techniques isolate tenants by providing separate VMs. This section describes security issues for VMM and VM respectively.

A. VMM Security

VMMs take an essential role for hypervisor and need to be trustworthy; yet they suffer from the following concerns.

1) VMM Vulnerabilities: VMMs cannot be immune to vulnerabilities since they are non-trivial pieces of software. Vulnerabilities of six major VMMs and emulators are investigated through source code auditing and fuzzing techniques [7]. All were found to have major flaws, leading to VMM abort and other exploitable areas. The presence of VMM vulnerabilities is further underscored by recent common vulnerabilities and exposures (CVE) [8] entries on, for instance, VMware and Xen. As with any other software, one should not render VMM as a single trustworthy piece of software.

2) VM-Based Rootkits: VM-based rootkits may take control of VMMs. While traditional kernel rootkits insulate themselves in the OS kernel, VM-based rootkits are in the VMM layer, thereby hiding from the kernel rootkit detector. Proof-of-concept VM-based rootkits were developed using Virtual PC and VMware, empirically observing the stealthiness of such an environment by measuring installation and boot times, along with memory footprints [9]. Another VM-based rootkit, Blue Pill [10], can be dynamically installed on the fly without requiring system reboot. Blue Pill can leverage the nested virtualization feature of AMD SVM, enabling it to insulate itself in the VMM layer while another hypervisor is running.

3) VMM Transparency: Another concern is the detectability of VMMs. It becomes especially evident in the context of hosting potentially hostile code, such as honeypot, which needs to thwart VM detection. [11] argued that the idea of building a transparent VMM is infeasible, based on speculation that various discrepancies between the physical machine and VM provide a number of clues: hardware abstraction, time sources, and overhead. [12] identified four quadrants of VMM detection problem space and run experiment with remote detection of VMM types. It detected the presence of a specific processor architecture (Pentium IV) and type of VMM (Linux, Xen, VMware) by a remote verifier. Pinpointing the type and version of VMM may lead to a renewed threat, since it may be possible to employ effective attacks against a specific VMM.

4) Platform Integrity: In multitenant environments, we tend to blindly trust underlying infrastructure such as VMM and the OS kernel. The chain of trust should in fact be assured at every layer of the software stack. Among various approaches is Terra [13], which is a prototypical trusted VMM that assigns different VMs for each application. Since each application may deploy optimized OS, the integrity of the security throughout the layers could be preserved. Another is the Trusted Platform Module (TPM), a security specification defined by Trusted Computing Group [14], which is being extended to accommodate virtualization techniques. A virtual TPM (vTPM) [15] is a software embodiment of the TPM. It can be run on an external co-processor card as well as on a VM. It extends the existing TPM 1.2 command set to accommodate vTPM, thus makes TPM accessible to every VM.

On the other hand, there exist a tool that measures the integrity of VMs running on top of hypervisor. One such tool is HIMA [16], a hypervisor-based agent. To measure integrity, it is required to isolate the measurement agent and measurement target, and to make sure that only the VMs that have passed the integrity check are running; thus only healthy programs are executed.

B. VM Security

On top of VMM are VMs, where tenants reside. Though VMs need to be secure and isolated among each other, this is not always the case.

1) Introspection: VM debugging and introspection techniques pose significant challenges to guest VMs under untrusted VMMs since they enable tracking of processing and data flows inside the guest VM, requiring no privilege inside it. One such implementation is MoonSols' LiveCloudKd [17], which debugs a guest OS from the hypervisor. LiveCloudKd allows us to run the KD (Windows kernel debugger) and WinDbg for introspecting guest Windows VMs from the Microsoft Hyper-V R2 hypervisor. Another such one is the VIX tool suite [18], which is a VM introspection system for Xen and can track processes of guest VMs from VMM by mapping a DomU virtual memory address into Dom0.

2) Redundancy Cancellation: Hypervisors usually isolate VMs, but complete isolation is not necessarily the best solution from the viewpoint of efficient resource utilization, and it is sometimes preferable to share resources among VMs without compromising security. sHype [19] enables resource sharing among VMs without compromising security. To do that, it enforces Mandatory Access Control (MAC)-based security policy. It does not cause significant processing overhead.

VI. OS-Layer Security

On top of VMs are OSs, which were invented decades ago, and whose security aspect has been researched until now. Only handful of studies are visible lately in this area since OS is an established technique. Among them are privilege separation and kernel integrity issues.

A. Privilege Separation

Privilege separation, or compartmentalization, controls access in order to separate user privileges. End systems must be capable of doing that based on confidentiality and integrity requirements to provide system security, and OS security mechanisms are the foundation for ensuring such separation.

Discretionary Access Control (DAC) is a conventional approach, which enables individual users to set up their own access policy. The DAC-based approach, however, is vulnerable, especially to Trojan horses and the exploitation of buggy software. As a consequence, application security mechanisms are vulnerable to tampering and bypass, and malicious or flawed applications can easily cause failures in system security.

To handle this, MAC-based approaches are studied. MAC enforces system policies: system administrators specify policies, which are checked via runtime hooks inserted into many places in the OS's kernel. Thus individual users cannot freely specify policies on their own files. Security Enhanced Linux [20] implemented MAC, and its architecture has been subsequently mainstreamed into Linux and ported to several other systems, including the Solaris OS, FreeBSD OS, and Darwin kernel. Another example is AppArmor [21], which also implemented MAC and supported Linux distributions including SUSE, PLD, Pardus Linux, Annvix, Ubuntu, and Mandriva. Tools exist for analyzing and comparing the quality of protection (QoP) offered by MAC systems. One such tool is VulSan [22]. It encodes security policies, system states, and system rules using logic programs and computes a host attack graph and the vulnerability surface when an attack scenario is given.

However, they are not capable of handling ambient authority, which results in enabling an application to fully access a user's data. To cope with the issue, Capsicum capabilities [23] were introduced. This is an extension of UNIX file descriptors and reflects rights on specific objects such as files or sockets, and may be delegated from process to process in a granular way in the same manner as other file descriptor types via inheritance or message-passing. This extension supports application compartmentalization, the decomposition of monolithic application code into components that will run in independent sandboxes to form logical applications.

B. Kernel Integrity

Exploitation of vulnerabilities in the OS kernel or inadvertent execution of untrusted software by a privileged user may lead to introduction of covert malware inside the kernel; i.e., a kernel rootkit. This could be prevented by employing VM introspection techniques. NICKLE [24] is a VMM feature extension for performing memory shadowing and runtime kernel code integrity checks. NICKLE is claimed to be a widely applicable technique for multiple VMMs, such as QEMU, VirtualBox, and VMware. NICKLE successfully defended the kernel against 23 real-world kernel rootkits.

VII. Application-Layer Security

Application-layer techniques provide logical resource isolation following a database for each user and data confidentiality. Recent such techniques are introduced below.

A. Secure Data Storing

Recent researches on securely storing user data and maintaining its usability are introduced below.

1) Encrypted Data Manipulation: The user's data must be confidential and protected from unauthorized access, and proposals have been made in that regard. Progressive elliptic curve encryption (PECE) [25] is a type of proxy reencryption [26], which allows a piece of data to be encrypted multiple times using different keys such that the final cipher text can be decrypted in a single run with a single key. With this mechanism, a data owner may encrypt data before uploading it to a service provider's server which, upon request, re-encrypts it by using the public key of the authorized party who wishes to access it, and the party receives and decrypts it with a single key. Note that the service provider knows no key to decrypt the data here, thus only the data owner can access the data. This enables secure usage of an untrustworthy service provider.

On the other hand, users may wish to manipulate their encrypted data stored at the provider's server (e.g., retrieve data) without decrypting that at the server in order to preserve confidentiality. Homomorphic encryption [27], [28] could achieve that. It is a form of encryption where a specific algebraic operation performed on the plain text is equivalent to another (possibly different) algebraic operation performed on the ciphertext. With this scheme, cloud service providers can perform complicated processing of data without being able to see it. This helps make cloud computing compatible with privacy.

2) Data Integrity: Techniques to ensure storage correctness across multiple servers or peers are proposed [29–31] to secure data integrity. Different from that, [32] designed a scheme that supports secure and efficient dynamic operations on data blocks, including data update, delete, and append. It allows the user to generate a homomorphic pre-computed token that is erasure-coded and stored locally. This is used to determine the misbehaving server.

3) Data Redundancy Cancellation: Deduplication, a technique that stores only a single copy of redundant data and provides links to that copy instead of storing other actual copies of the data, is now widely used by cloud storage providers to efficiently store data. Yet such deduplication may cause security risks. [33] pointed out the potential risks of cross-user source-based deduplication and described how such deduplication can be used as a side channel to reveal information about the contents of other users' files, and as a covert channel by which malicious software can communicate with the outside world, regardless of the firewall settings of the attack machine. To handle this, the article discussed three candidate schemes: using encryption to prevent deduplication, performing deduplication at the server, and setting a randomized threshold for running deduplication. Though these are not perfect solutions, they provide higher privacy guarantees while slightly reducing the benefit of deduplication.

B. User Data Isolation

Individual users' data needs to be isolated to preserve privacy. Two major approaches for that are introduced below.

1) Process Isolation: Data in a multitenant system needs to be carefully treated so that it cannot be mixed with another user's. Information flow control (IFC) [34] tags data entering the system and isolates the data belonging to different users. [35] implements IFC constraints in the Erlang programming language by leveraging its shared-nothing computation, message passing, and lightweight processes to provide uniform privacy preservation in a highly concurrent application.

2) Selective Decryption Techniques: User data needs to be read by the intended users, but must not be read by the others. To enable that, cryptographic techniques enabling selective decryption have been researched. One fundamental technique enabling that is the identity-based encryption scheme [36], which enables a user specified by an identity to decrypt data. [37] designed a mutual authentication system that allows users in the same domain to securely share data for cloud storage by using the technique. There could be benefit in endowing decryption rights to a group of users instead of a single user designated by an identity, and [38] implemented a hierarchical identity-based architecture based on the hierarchical identity-based encryption (HIBE) system proposed by Gentry et al [39]. It predefines user structure, and decryption keys are created following the structures. Attribute-based encryption (ABE) provides more flexible decryption key assignment, and [40] defined access structures of files so that a user can access a file only if the user's attributes satisfy the file's access structure. Note that the original ABE is proposed by [41], extensions of which are available in [42–46]. By combining assorted techniques including above, [47] introduced an architecture for a cryptographic storage service that assumes untrusted service providers.

C. Authentication and Authorization

Data in the cloud need to be accessed by authorized parties. For that, authentication and authorization mechanisms are needed. Though this has been already studied for many years, many researches designed and tuned for cloud computing are actively conducted lately, which are introduced below.

1) Authentication Framework: The authentication and authorization model could be refined in the multitenant cloud computing environment, and researches proposing such models are reported. [48] designed an authorization system suitable for middleware service in PaaS. The system is based on a model defining a 5-tuple (Issuer, Subject, Privilege, Interface, Object), which is incorporated with role-based access control (RBAC), hierarchical RBAC (hRBAC), path-based object hierarchies and federation, OpenID, and X.509 to make the system robust. During an authorization request, it uses all this information to determine if the request is authorized.

2) Query Control: Though access control permits or denies access to a resource based on a query, the control could be more finely tuned by investigating the query. One such tuning is enabled by a Declarative Secure Distributed Systems (DS2) framework [49]. This provides secure query processing in a multi-user cloud environment and rewrites the query when needed. Note that it also enables seamless integration of declarative access control policies with data processing to enable secure sharing among users. Another such tuning is proposed by [50], which introduced a system, in which customers access databases through a privacy-enabling engine that enforces policies expressed using a semantic Web language that allows customers to generate compliant queries. If a customer's query violates the policy, the policy reasoner outputs a justification. Semantic Web language offers assurance that humans and machines can attain common understanding on policies and generates justifications that are human-readable.

VIII. Web Security

Web applications are one type of application that is discussed above. On the other hand, Web is inevitable for multitenant services since most of them, especially public cloud computing services, use the Web as a means for users to access them. It thus needs to be secured.

A. Server-side Security

Web applications are often multitenant, and need to be protected from being subverted to keep other tenants from being victims of a subverted system or malicious tenant. Major researches on server-side security are introduced below.

1) Content Usage Control: Web pages naturally use or refer to external contents. This increases the modularity and usability of Web contents, and is an essential feature of Web applications. However, that feature, mashup, becomes a source of vulnerability. Mashup services are created from the aggregation of services from different origins. While the owner of an application typically wishes that contents abide by the Same-Origin Policy (SOP), mashup services sometimes require cross-domain communication.

To cope with that, Smash [51] considers contents from different origins as different components running in the browser window. Components can be loaded and unloaded dynamically and communicate with each other using communication channels. This channels are monitored and controlled by event hub, which applies security policies. Another approach is Omash [52], which treats each web page as an object and allow the object to communicate only via its declared public interfaces.

In addition to communication control mechanism, content usage policy within a site could be defined to control external content usage. Content Security Policy (CSP) [53] is a content restriction enforcement scheme that allows developers to specify how content interacts on their websites. A policy is provided to the site via HTTP, specifying what may load onto the site and into what context. User input is also protected and only accessed by the site and its authorized users. Its test implementation is available as a browser plug-in.

2) Access Control Vulnerabilities: Access control attacks occur when the access control in a Web application is incorrect or missing, allowing unauthorized access to privileged resources. In multitenant environments, a tenant with low privilege may be able to gain higher permissions if authorization is not correctly enforced. Still, audit facilities can deter an authenticated tenant from attempting to do harm. Typical authentication schemes are login-password based, but recent studies attempt to abandon such practices.

An `Origin` header [54] provides the origin in the sense of the SOP, and does not leak information contained in the URI that is usually provided by the `Referrer` header. The `Origin` header is only usable via POST requests. The header preserves the user's privacy, but this requires modification of browsers to implement the header and accommodate HTTP transactions fielding it. Developers are also required to enforce best practices in the usage of GET and POST requests in that POST should always be used for state-modifying requests while state-modifying GET requests should be blocked.

Nemesis [55] is a system that automatically tracks the flow of user credentials when authentication is performed. It generates an additional HTTP cookie to track requests from an authenticated user by dynamic information flow tracking (DIFT) and runs shadow authentication with this information to enforce developer-specified access control rules. It requires no modification to the application, either to the authentication or its access control system. DIFT can reduce false positives and improve the precision of real-world security tools.

B. Client-Side Validation

Many of current Web applications consist of server-side and client-side modules. Vulnerabilities on client-side modules are often caused by the unsafe use of untrusted code within a software. These vulnerabilities are called client-side validation (CSV) vulnerabilities, and many codes causing CSV vulnerabilities are JavaScript. Researches on detecting the vulnerabilities in advance have been reported.

1) Endpoint Risk Detection: Endpoint risk detection techniques detect client-side vulnerability at the endpoint. Many of the techniques including FLAX [56] and Zozzle [57] target JavaScript issues. FLAX is a system that systematically discovers CSV vulnerabilities by combining the features of dynamic taint analysis with those of automated fuzz testing to generate test cases that concretely demonstrate the presence of the vulnerabilities. Dynamic taint information extracts knowledge of the type of sink operation involved in the vulnerability, thereby specializing the subsequent black box fuzzing for each sink type. This method eliminates false alarms that would result from a purely taint-based tool. Zozzle is a mostly static JavaScript Malware detector that focuses on heapspray attacks. It is a Bayesian classifier that learns features on an abstract syntax tree (AST) of the JavaScript Malware deobfuscation. These features represent a hierarchical structure of context (generated during deobfuscation), both benign and malicious, in order to understand which context leads to which context. It enjoys efficient AST-based feature extraction and fast pattern matching. The scheme is integrated into the JavaScript engine.

2) Middlebox Risk Detection: Different from above, this middlebox risk detection approach requires no modification to the client. Web content is investigated, processed at the middlebox, then transfered to the endpoint so that the endpoint will not be exposed to harmful Web content. Webshield [58] relies on segregating the Web application code into HTML and non-HTML content. HTML content processing has two main steps: an initial page transformation that encodes possible untrusted data injection points and a dynamic HTML interaction support that will run JavaScript contents in a sandbox browser and reflect updates to the real browser in places that were encoded. It enjoys reasonable overhead in terms of communication, time, and memory, but it is vulnerable to non-deterministic execution of non-cacheable contents upon second download and cannot handle excessive DOM updates. Other such reports are BrowserShield [59] and SpyProxy [60] though they are not elaborated on here in the interest of space.

IX. Considerations and Discussion

This paper identified technical issues on multitenant cloud computing, which are summarized in Table II. Based on the table, this section discusses the technical maturity of multitenant cloud computing and the needs for advancing operational security.

A. Technical Maturity

The technical maturity, in terms of security, differs among the technical layers. As can be seen from the above survey results, some layers have various active researches while others do not, and some layers have solution proposals to cope with security issues while others only identify security issues. HW&SW primitive techniques have been researched for decades, but the issues are new, and not so many people could handle the techniques in this layer; thus it was difficult to find any article on countermeasures to the issues. Hypervisor techniques are rather new, and so much research focuses on their development in terms of functionality and performance rather than security. Researches on OS-layer techniques are cooling since they have already been researched for decades; it was not easy for us to identify articles discussing the resource isolation issues in this layer. Application-layer techniques are still evolving, and many articles on issues and countermeasures are found; many of the issues are applicable to non-multitenant cloud computing, they presented new features for multitenant cloud computing. Web techniques are also attracting diverse research since they are critical topics for the emerging SaaS business. Above disparities come from the differing technical maturity among techniques in each layer.

Various issues apparently need to be tackled toward the practical security of multitenant cloud computing services, and these are often claimed to be the reasons for not using such services. Nevertheless, as discussed above, these issues arise partly due to the current stage of technical maturity, and it is unfair to say at this stage that multitenant cloud computing is characteristically insecure. Though techniques in some layers have very few countermeasures at this moment, the situation will be improved gradually as they approach maturity.

B. Needs for Operational Security

As discussed above, security techniques on multitenant cloud computing are not yet mature and need to be developed further. Even if the techniques are reaching maturity, users never get guarantees on security, and humans remain a point of vulnerability [61], [62]. One approach coping with that is implementing adequate cybersecurity operations that needs to be taken before and after incidents. Though basic operations do not change, cloud computing requires further complication on the operations [63].

Many of the users of multitenant cloud computing services are, however, non-familiar with such operations. In case of

TABLE II
MAJOR SECURITY-RELATED CHALLENGES FOR MULTITENANCY

Layers	Categories	Issues	References
Web	Server-side security	Content usage control	Component access control (Smash/Omash) [51], [52] Content security policy [53]
		Access control vulnerabilities	Origin header [54] Nemesis: shadow authentication [55]
	Client-side validation	Endpoint risk detection	FLAX: CSV vulnerability scanning [56] Zozzle: JavaScript malware detection [57]
		Middlebox risk detection	Proxying and auditing communication [58–60]
Application	Secure data storing	Encrypted data management	Progressive elliptic curve encryption (PECE) [25] Homomorphic encryption [27], [28]
		Data integrity	Assuring storage correctness [29–32]
		Redundancy cancellation	Cross-user deduplication [33]
	Authentication and authorization	Authentication framework	Authentication model for multitenant environment [48]
		Query control	DS2 framework providing query rewriting [49] Semantic policy [50]
	User data isolation	Process isolation	Information flow control [35]
		Selective decryption techniques	(Hierarchical) ID-based encryption [36–39] Attribute-based encryption [40–46] Cryptographic cloud storage service architecture [47]
Operating System	Privilege separation		MAC-based separation [20], [21] Capsicum: capability-based separation [23] VulSan: QoP analyzer [22]
	Kernel integrity		NICKLE: kernel rootkit prevention [24]
Hypervisor	VMM security	Vulnerabilities	Existing VMM's vulnerabilities [7]
		Rootkits	VM-based rootkits [9] [10]
		Transparency	VMM-indicative discrepancy information [11] VMM detection experiment under Pentium IV [12]
		Platform integrity	Building trustworthy platforms [13], [15] Measuring platform integrity [16]
	VM security	Introspection	VM introspection for Xen [18]
		Redundancy cancellation	sHype: MAC-based security architecture [19]
HW&SW Primitive	Programming code	Managed runtime security	Java and .net vulnerabilities [1], [2] Java introspection [3]
		Sandboxing	User-level sandboxing [4]
	Hardware processing	Shared processor resources	Side channels in Pentium 4 [5] Intel architecture problem [6]

cloud computing, the matter is exacerbated due to its resource combination nature, which requires complicated configurations and leads to misconfigurations that are vulnerable to attacks. Thus these operations had better be automated or semi-automated to improve operation efficiency and to avoid any human incidents, and various such activities are already currently available [64–67]. Along with the multitenancy techniques, cybersecurity operations need to be advanced further toward secure multitenant cloud computing environment.

X. Summary and Conclusion

This paper introduced technical layers and categories, with which it identified and structured security challenges and approaches of multitenant cloud computing. The survey was a snapshot of security issues of multitenant cloud computing, and mainly focused on the issues related to resource isolation techniques. Based on the survey, this paper discussed the differing technical maturity among technical layers; some layers such as hypervisor and Web are still evolving, and their security techniques have yet to be improved further, while others are approaching a certain technical maturity. Thus it is unfair to say at this moment that cloud computing services are characteristically insecure. To use cloud computing services, users need to consider operational security. Security operations and their automation need to be researched further to improve their efficiency and to alleviate security risks. Tackling security issues from both technical and operational aspects will eventually expedite wider deployment of multitenant cloud computing.

References

[1] E. Metula, "Managed Code Rootkits: Hooking into Runtime Environments," in *BlackHat USA*, 2009.

[2] E. Metula, ".NET Framework Rootkits: Backdoors Inside Your Framework," in *BlackHat Europe*, 2009.

[3] A. Dabirsiaghi, "JavaSnoop: How to hack anything in Java," in *BlackHat Las Vegas*, 2010.

[4] B. Ford and R. Cox, "Vx32: lightweight user-level sandboxing on the x86," in *USENIX ATC*, 2008.

[5] C. Percival, "Cache missing for fun and profit," in *BSDCan*, 2005.

[6] J. S. Robin and C. E. Irvine, "Analysis of the intel pentium's ability to support a secure virtual machine monitor," in *USENIX Security Symposium*, 2000.

[7] T. Ormandy, "An Empirical Study into the Security Exposure to Hosts of Hostile Virtualized Environments," in *CanSecWest*, 2007.

[8] The MITRE Corporation, "Common Vulnerability and Exposures (CVE)," http://cve.mitre.org/, Mar. 2011.

[9] S. King and P. Chen, "Subvirt: implementing malware with virtual machines," in *IEEE Symposium on Security and Privacy*, May 2006.

[10] J. Rutkowska, "Subverting Vista kernel for fun and profit," 2006.

[11] T. Garfinkel, *et al.*, "Compatibility is not transparency: Vmm detection myths and realities," in *HotOS*, 2007.

[12] J. Franklin, *et al.*, "Remote detection of virtual machine monitors with fuzzy benchmarking," *SIGOPS Oper. Syst. Rev.*, April 2008.

[13] T. Garfinkel, *et al.*, "Terra: a virtual machine-based platform for trusted computing," in *SOSP*, 2003.

[14] Trusted Computing Group, http://www.trustedcomputinggroup.org/, June 2011.

[15] S. Berger, *et al.*, "vTPM: virtualizing the trusted platform module," in *USENIX Security Symposium*, 2006.

[16] A. Azab, *et al.*, "Hima: A hypervisor-based integrity measurement agent," in *ACSAC*, dec. 2009.

[17] Moonsols, "LiveCloudKd," *http://www.moonsols.com/2010/08/12/livecloudkd/*, Aug. 2011.

[18] B. Hay and K. Nance, "Forensics examination of volatile system data using virtual introspection," *SIGOPS Oper. Syst. Rev.*, April 2008.

[19] R. Sailer, *et al.*, "Building a mac-based security architecture for the xen open-source hypervisor," in *ACSAC*, 2005.

[20] National Security Agency, "Security enhanced linux," *http://www.nsa.gov/research/selinux/*, May 2011.

[21] Novell Inc., "Apparmor application security for linux," *http://www.novell.com/linux/security/apparmor/*, May 2011.

[22] H. Chen, N. Li, and Z. Mao, "Analyzing and comparing the protection quality of security enhanced operating systems," in *NDSS*, 2009.

[23] R. Watson, *et al.*, "Capsicum: practical capabilities for unix," in *USENIX Security*, 2010.

[24] R. Riley, X. Jiang, and D. Xu, "Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing," in *RAID*, 2008.

[25] G. Zhao, *et al.*, "Trusted data sharing over untrusted cloud storage providers," in *CloudCom*, Nov. 2010.

[26] M. Masahiro and O. Eiji, "Proxy cryptosystems: Delegation of the power to decrypt ciphertexts," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, Jan. 1997.

[27] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, 2009.

[28] C. Gentry, "Computing arbitrary functions of encrypted data," *Commun. ACM*, March 2010.

[29] T. Schwarz and E. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," 2006.

[30] M. Lillibridge, *et al.*, "A cooperative internet backup scheme," in *USENIX ATC*, 2003.

[31] K. Bowers, A. Juels, and A. Oprea, "Hail: A high-availability and integrity layer for cloud storage," in *CCS*, 2009.

[32] C. Wang, *et al.*, "Ensuring data storage security in cloud computing," in *IWQoS*, July 2009.

[33] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *Security Privacy, IEEE*, Nov. 2010.

[34] A. C. Myers and B. Liskov, "Protecting privacy using the decentralized label model," *ACM Trans. Softw. Eng. Methodol.*, Oct. 2000.

[35] I. Papagiannis *et al.*, "Enforcing User Privacy in Web Applications using Erlang," in *W2SP*, 2009.

[36] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO*, 2001.

[37] L. Kang and X. Zhang, "Identity-based authentication in cloud storage sharing," in *MINES*, Nov. 2010.

[38] Q. Liu, G. Wang, and J. Wu, "Efficient sharing of secure cloud storage services," in *CIT*, June 2010.

[39] C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography," *Advances in Cryptology. ASIACRYPT*, 2002.

[40] J. Li *et al.*, "Fine-grained Data Access Control Systems with User Accountability in Cloud Computing," in *CloudCom*, 2010.

[41] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT*, 2005.

[42] V. Goyal, *et al.*, "Attribute-based encryption for fine-grained access control of encrypted data," in *CCS*, 2006.

[43] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in *IEEE Symposium on Security and Privacy*, May 2007.

[44] M. Chase, "Multi-authority attribute based encryption," *Theory of Cryptography*, 2007.

[45] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *CCS*, 2009.

[46] S. Yu, *et al.*, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *IEEE INFOCOM*, Mar. 2010.

[47] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *FC*, 2010.

[48] J. Calero, *et al.*, "Toward a Multi-Tenancy Authorization System for Cloud Services," *Security Privacy, IEEE*, Nov. 2010.

[49] W. Zhou, *et al.*, "Towards a data-centric view of cloud security," in *CloudDB*, 2010.

[50] L. Kagal and J. Pato, "Preserving Privacy Based on Semantic Policy Tools," *IEEE Security & Privacy*, 2010.

[51] F. De Keukelaere, *et al.*, "SMash: Secure Component Model for Cross-Domain Mashups on Unmodified Browsers," in *WWW*, 2008.

[52] S. Crites, F. Hsu, and H. Chen, "OMash: Enabling Secure Web Mashups via Object Abstractions," in *CCS*, 2008.

[53] S. Stamm, B. Sterne, and G. Markham, "Reining in the Web with Content Security Policy," in *WWW*, 2010.

[54] A. Barth, C. Jackson, and J. C. Mitchell, "Robust Defenses for Cross-Site Request Forgery," in *CCS*, 2008.

[55] M. Dalton, C. Kozyrakis, and N. Zeldovich, "Nemesis: Preventing Authentication & Access Control Vulnerabilities in Web Applications," in *USENIX Security Symposium*, 2009.

[56] P. Saxena, *et al.*, "FLAX: Systematic Discovery of Client-side Validation Vulnerabilities in Rich Web Applications," in *NDSS*, 2010.

[57] C. Curtsinger, *et al.*, "Zozzle: Low-overhead Mostly Static JavaScript Malware Detection," Microsoft Research, Tech. Rep., Nov. 2010.

[58] Z. Li *et al.*, "WebShield: Enabling Various Web Defense Techniques without Client Side Modifications," in *NDSS*, 2011.

[59] C. Reis, *et al.*, "Browsershield: vulnerability-driven filtering of dynamic html," in *OSDI*, 2006.

[60] A. Moshchuk, *et al.*, "Spyproxy: execution-based detection of malicious web content," in *USENIX Security Symposium*, 2007.

[61] Mark Fossi, et al., Ed., *Symantec Internet Security Threat Report*, vol. 16, Symantec Corporation, Apr. 2011.

[62] The Open Web Application Security Project, http://www.owasp.org, June 2011.

[63] T. Takahashi, Y. Kadobayashi, and H. Fujiwara, "Ontological approach toward cybersecurity in cloud computing," in *SIN*, 2010.

[64] A. Rutkowski, *et al.*, "CYBEX – The Cybersecurity Information Exchange Framework (X.1500)," *CCR*, Oct. 2010.

[65] T. Takahashi, *et al.*, "IODEF-extension to support structured cybersecurity information," *IETF Internet Draft (draft-ietf-mile-sci-03.txt)*, Feb. 2012.

[66] ICASI, "The Common Vulnerability Reporting Framework (CVRF) v1.0," http://www.icasi.org/cvrf, Aug. 2011.

[67] D. Waltermire, *et al.*, "The Technical Specification for the Security Content Automation Protocol (SCAP) : SCAP Version 1.2," *NIST Special Publication 800-126 Revision 2*, Sept. 2011.