Accepted Manuscript

Analyzing the Security of Windows 7 and Linux for Cloud Computing

Khaled Salah, Jose M. Alcaraz Calero, Jorge Bernal Bernabé, Juan M. Marín Perez, Sherali Zeadally

PII: S0167-4048(12)00180-0

DOI: 10.1016/j.cose.2012.12.001

Reference: COSE 663

To appear in: *Computers & Security*

Received Date: 5 April 2012

Revised Date: 28 August 2012

Accepted Date: 1 December 2012

Please cite this article as: Salah K, Alcaraz Calero JM, Bernabé JB, Marín Perez JM, Zeadally S, Analyzing the Security of Windows 7 and Linux for Cloud Computing, *Computers & Security* (2013), doi: 10.1016/j.cose.2012.12.001.

Analyzing the Security of Windows 7 and Linux for Cloud Computing

Khaled Salah
Electrical and Computer Engineering Department
Khalifa University of Science Technology and Research (KUSTAR)
PO Box 573, Sharjah, UAE
Email: Khaled.salah@kustar.ac.ae

Jose M. Alcaraz Calero
Department of Computer Science, Engineering Technical School
University of Valencia
46100 Valencia, Spain
Email: jose.m.alcaraz@uv.es

Jorge Bernal Bernabé, Juan M. Marín Perez
Cloud and Security Lab
Hewlett-Packard Laboratories
Stroke Gifford
BS34 8QZ Bristol, United Kingdom
Email: jorge.bernal@hp.com

Sherali Zeadally
Department of Computer Science and Information Technology
University of the District of Columbia
Washington DC, 20008, USA
Email: szeadally@udc.edu

Abstract

We review and analyze the major security features and concerns in deploying modern commodity operating systems such as Windows 7 and Linux 2.6.38 in a cloud computing environment. We identify the security weaknesses and open challenges of these two operating systems when deployed in the cloud environment. In particular, we examine and compare various operating system security features which are critical in providing a secure cloud. These security features include authentication, authorization and access control, physical memory protection, privacy and encryption of stored data, network access and firewalling capabilities, and virtual memory.

KEYWORDS: Cloud Computing, Operating System Security, Privacy, Security

1. Introduction

Cloud computing is emerging as a new attractive computing paradigm tailored to meet most of today's Information Technology (IT) business needs. It is based on the offering of virtual IT infrastructures as a service through the Internet. The idea is to enable businesses to rent part of their required infrastructures using a pay-as-you-go model in which the number of virtual devices is dynamically scaled to meet the current business demands. Cloud computing describes a logical stack divided into three different layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The IaaS layer is responsible for providing the virtual infrastructure (virtual machines, volumes, networks, etc.). The PaaS layer is in charge of providing middleware services which may be seen as value-added services. Likewise, the SaaS layer offers software features and services to end users, making use of the underlying services provided by PaaS layer.

IaaS cloud providers use different cloud solutions such as Openstack, Cloudstack or Amazon EC2 to provide a scalable environment. These solutions use hypervisors (such as Xen, KVM or VMWare) to offer virtualization and schedule access of guest Operating Systems (OSs) to physical resources such as CPU, memory or disk I/O. User Virtual Machines (VMs) instances are booted from OS images which are typically customized to be deployed in the cloud. These images can be either provided and configured by final users or chosen from a set of OS images offered by the cloud provider. VMs run on the physical infrastructure of the cloud provider, and deliver storage, computing and networking capabilities. Moreover, these VMs are rented to different customers and it is possible for two or more customers to share VMs running on the same physical infrastructure often referred to as multi-tenancy in cloud environments. Figure 1 shows a simplified view of a multi-tenancy scenario in which a cloud provider's physical computer (host) is virtualized and the virtual machines are rented to third-parties. The figure shows how two different cloud customers use different VMs while sharing the same cloud provider infrastructure.

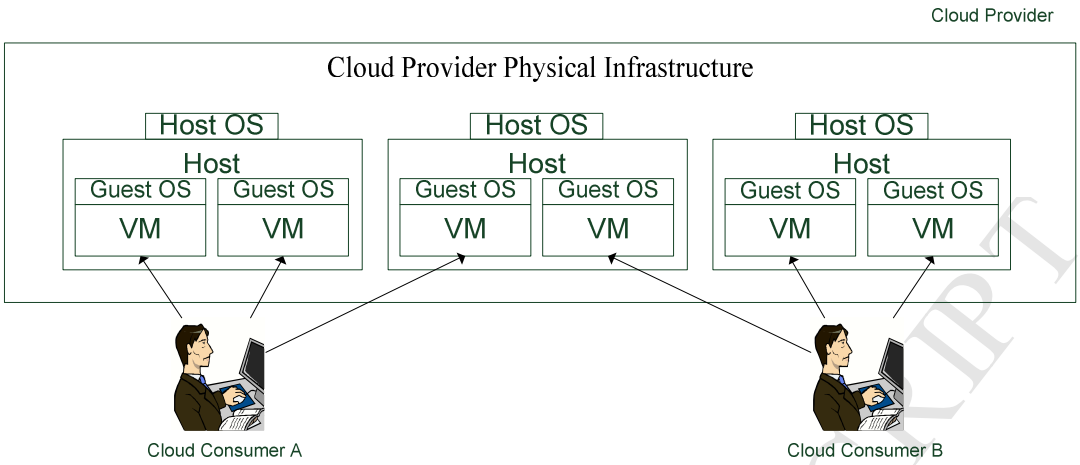

Figure 1 - Simplified multi-tenancy cloud scenario

From the customer perspective, the usage of third-party virtual infrastructures entails an important security challenge. The cloud provider potentially has full control over its own machines whereas the customer wants to keep his/her security and privacy in the rented infrastructure. Security and privacy are not only required for the external users of such an infrastructure but are also needed for the cloud infrastructure provider. These security and privacy issues still need to be addressed today for the cloud environment.

VMs usually run a guest OS such as Windows-based and Unix-based. These guest OS images run in the VMs in the cloud provider's infrastructure. Generally, most cloud providers offer a set of predefined and preconfigured OS images and make them available to customers. Some cloud providers do not offer the capability to upload bootable disk images while others support this capability. In the latter case, the cloud provider has full access to the guest OS for provisioning "security information" within the guest OS image. In both cases, a trust relationship between cloud provider and customer is enforced. This type of trust relationship is probably one of the main causes that prevent customers from using the cloud, particularly if the security of cloud provider gets compromised. In this paper, we explicitly try to avoid this forced trust relationship by assuming no trust relationship exists between the cloud customer and the cloud provider.

Contributions of this work

The main contribution of this paper is a comprehensive security analysis of today's operating systems when executed as VM images in the cloud which is a significantly different execution environment than the traditional execution environment for which these OSs were originally designed for. This analysis may be critical to enable cloud customers to identify the open challenges and weaknesses of the main current OSs when they run in the cloud environment. We discuss and analyze the security

features and solutions offered by the most popular commodity Operating Systems namely, the latest versions of Windows (*Windows 7 Ultimate Edition)* and Linux (*Fedora Core 15* with 2.6.38 kernel version) with a special attention paid to their suitability for the cloud computing environment, especially when no trust relationship between cloud provider and customer is assumed. There are indeed many security features and aspects to be analyzed. This study does not attempt to cover all of them. We only focus on those aspects which can affect the security of a guest operating system running in a VM in a cloud environment. Our main goal is to identify security and privacy features of current OSs that are not suitable for use in a cloud computing environment. In addition, we identify important security aspects which may be leveraged to establish a secure running environment of the guest OSs when running in the cloud.

This rest of this paper is organized as follows. Sections 2, 3, 4 and 5 analyze respectively authentication, authorization, accounting and privacy capabilities of Windows and Linux operating systems when used in cloud computing environments. We identify security open issues, strengths, weaknesses, and concerns associated with these aforementioned capabilities. Section 6 discusses why current OSs are not suitable for use in a cloud computing environment. Finally, Section 7 concludes the study and presents future work.

2. Authentication

Authentication is the process by which the OS verifies that a user is really who he claims to be. Usually, the user is prompted for some credentials which are then checked to verify his identity. There are different types of credentials such as password, certificates, or biometrics, as well as different authentication mechanisms that verify these credentials.

In Windows 7 OS, authentication is mainly performed by the *Local Security Authority* (LSA) authentication component (Microsoft, 2003). Applications can make use of this component to authenticate users in the local system. LSA authentication relies on *Authentication Packages* (AP) and on *Security Support Providers* (SSP) (Leach, 2003) to perform its tasks. APs implement different authentication mechanisms to perform the logon in the operating system whereas SSPs implement different security protocols to enforce a secure authentication environment. Currently, Microsoft provides at least two APs: *MSV1_0 AP* checking the user/password data either against the *Security Account Manager* (SAM) database (locally stored and in the domain controller) and *Kerberos AP* (Neuman, 2005) used to authenticate a user in a network using the Kerberos ticket-based protocol. SSPs enable the design of an integrated authentication architecture by allowing applications to make use of different security protocols without the need to change their interfaces. Currently, the following SSPs are provided by Windows 7: *i) NT Lan Manager NTLM* (Microsoft NTLM, 2011) *protocol;*

ii) *Kerberos*: Kerberos protocol. iii) *Credential Security Support Provider (CredSSP)* which uses Transport Layer Security (TLS) protocol and the *Simple and Protected Negotiate* (SPNEGO) protocol to authenticate the user with either NTLM or Kerberos. iv) *Digest SSP*: A lightweight authentication protocol using Hypertext Transfer Protocol (HTTP) or Simple Authentication Security Layer (SASL) (A. Melnikov and K. Zeilenga, 2006) protocol.

In contrast, Linux authentication is mainly based on the *Pluggable Authentication Modules* (Samar, 1996) (PAM). These modules provide an abstraction layer for the authentication services supported in the system. The modules are developed as libraries which are loaded dynamically. There are several PAM modules provided in Linux distributions. Some of the common modules implement authentication mechanisms and are usually included with PAM are: i) *pam_krb5*: Kerberos 5 protocol. ii) *pam_radius*: Remote Authentication Dial In User Service (RADIUS) protocol. iii) *pam_guest*: authorizes users as guests using fixed usernames. iv) *pam_unix*: traditional Unix login and password authentication mechanism.

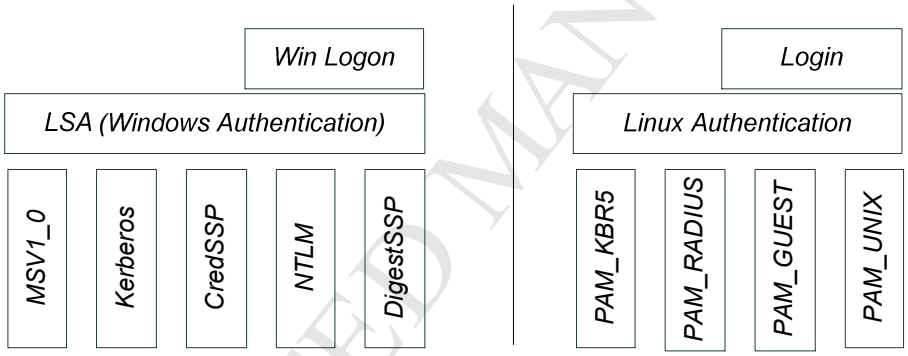

Figure 2. Authentication Architecture in Windows (left side) and Linux (right side)

Figure 2 shows an overview of the different authentication architectures provided by Windows and Linux. Regarding the login process, in Windows, the *Winlogon* module is in charge of interactively authenticating users when logging into the operating system. This module makes use of credential providers in charge of gathering actual user credentials. *Winlogon* passes these credentials to the *LSA* to validate them and perform the authentication. In Linux systems, the default application to authenticate users is the *login* application. This application makes use of the PAM library to authenticate users.

In this analysis, we have considered cloud computing solutions that allow cloud customers the option to upload their own guest OS image. The other cloud computing solutions usually need to modify the OS image in order to configure the authentication. The most frequent approach is enabling user access through SSH using certificate keys, which is used in OpenStack, one of the most widely used IaaS

implementations. A public-private key pair is issued by the cloud provider for each user. This key pair is used with the OS image file system before booting it when the instance is created. However, this approach requires the cloud provider to have access to the OS image. This implies a high security threat because the cloud provider could potentially access tenant information contained in the VM. Moreover, file system encryption cannot be used with this approach to protect the information because that would disallow the insertion of user keys. Thus, an enhanced mechanism is required to enable the provisioning of tenant user accounts into the VM to avoid access to the OS image by the cloud provider.

As can be seen from the previous analysis, both Windows and Linux provide abstraction mechanisms that enable applications to authenticate users, while being independent of the underlying authentication model. This enables the authentication information to be kept out of the OS in a dedicated subsystem. The use of an Identity Management (IdM) system and its integration into the underlying OS would enable the provisioning of user accounts without requiring access to the image file system. Otherwise, the guest OS may be attacked by brute force or dictionary techniques in order to retrieve the authentication information from local stores when the VM is powered off. The IdM handles user accounts and the OS makes use of this information. The separation of authentication from the underlying image file system also means that almost all the authentication modules previously introduced for both Windows and Linux OSs do not fit in cloud computing. Those modules related to local storage of credentials are discarded and those modules related to remote authentication protocols may resist against man-in-the-middle attacks. Thus, the authentication exchange protocol has to be secure enough to resist such kind of attacks. With the integration of Kerberos, both OSs can be effective against man-in-the-middle attacks. More advanced IdM solutions, such as those using the SAML or OpenID specifications, may provide enhanced capabilities. Both operating systems provide extensible mechanisms that can be used to integrate with these IdM solutions to provide an enhanced user provisioning solution for new VM instances.

However, the inclusion of IdM should be handled and managed by the tenant organization. If the cloud provider is providing or managing the IdM system, it might be able to gain access to the OS information because it will have control over the user accounts for the VM. Moreover, the communication between the OS and the IdM system should be secured and authenticated. That is, the OS should be able to verify that it is relying on the legitimate IdM system of the corresponding tenant. This situation introduces a new challenge regarding the provisioning of the credentials or certificates to enable this verification without the need to modify the OS file system for each instance by the cloud provider.

Another problem regarding authentication is brute force attack for password or credentials. In a virtualized environment such as cloud computing, we must distinguish between two different situations. The first one is when the VM is powered off. In this situation, if the attacker is able to access the file system, it can get access to the credential information (e.g., passwd file in Linux systems) and make use of brute force attacks later to get access to the VM when it gets powered on. The second situation is when the VM is powered on. In regular deployments, this is addressed by some blocking mechanism that prevents further login when a certain number of attempts is reached. But this mechanism is not feasible in a cloud computing environment because the OS is being executed in a virtualized environment. Thus, multiple instances of the same image can be run and the attacker can run a new instance when the maximum number of attempts is reached. The problem is exacerbated when the attacker has the ability of snapshotting the VM. This means that the attacker is able to resume the execution of the VM from the login prompt as many times as he wants or even run different instances in parallel. Thus, it reduces the time to have another set of login attempts. It also allows the attacker to use any desired VM image instance along with the desired set of information to be compromised.

The inclusion of an IdM system also protects against credential brute force attacks. When the VM is powered off, the credential information cannot be accessed even if the attacker has access to the file system. This is because credentials are located in the IdM system instead of being stored in the VM file system. In addition, the inclusion of an IdM system also enables to limit the number of login attempts. Thus, it can be used to protect against brute force attacks when the VM is running.

Finally, another open issue is the protection against threats when the runtime information used by the OS to contact the IdM system can be changed by a fake service in order to gain access to the VM. This threat can only happen if the attacker gains access to the file system or the RAM memory. This situation can be addressed with data privacy methods we describe in section 6.

3. Authorization

Access control in operating systems manages the privileges granted for a given authenticated user and checks and enforces such privileges when the user tries to perform actions over a set of securable objects. Windows 7 provides different access control subsystems which are executing simultaneously at run-time. Firstly, it implements an access control subsystem to control all the resources managed in the OS. An overview of the authorization model executed by this subsystem is shown in Figure 3. In essence, a *securable object* is defined as any object which may be protected, for example, a *file, directory, process, registry key, windows service, printer,* etc. Each securable object is associated with a *security descriptor* which contains the security information related to that object. The security

descriptor manages two different access control lists: *Discretionary Access Control List* (*DACL*) and *System Access Control List* (*SACL*). The *DACL* is responsible for the actual object access control while the *SACL* is in charge of audit related tasks. These lists are composed of a set of ordered *access control entries*. These entries determine the *access rights* granted to each particular *user* or *group* for the *securable object* protected. When users are authenticated into the system, a new *access token* is created, containing the identifier of the authenticated user and all its associated groups. It is associated with all the processes created by that user. The access control mechanism is invoked when a process attempts to access a *securable object*.

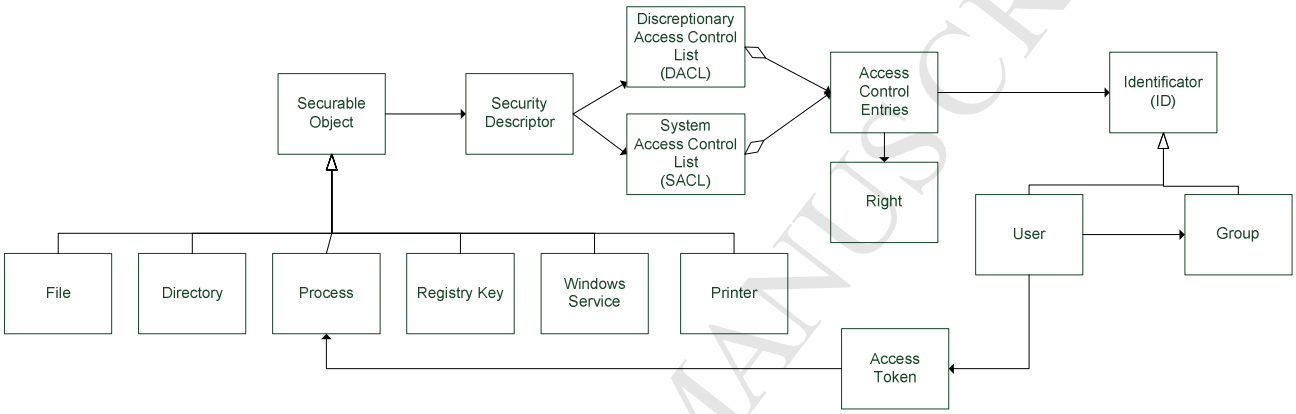

Figure 3. Discretionary Access Control Model implemented in Windows 7

Windows 7 also implements an extra access control for controlling the access to *securable objects*. This is referred to as the *Mandatory Integrity Control* (Conover, 2010) (Riley) *(MIC)* and it is an addition to the discretionary access control. Essentially, each user and securable object is associated with a security level. This security level determines the levels of protection assigned to the user and to the securable object. Four different security levels are defined in Windows 7: *low, medium, high* and *system*. Only those users and processes which have a higher security level than the one required by a *securable object* are able to access such an object. Otherwise, any access to the *securable object* is denied even if the other authorization system allows such an access.

The third access control subsystem implemented in Windows 7 is *AppLocker*. It controls the execution of *processes*. When a user executes an application (or installer), there is a Windows service (called the *Application Identity* service) that intercepts this execution. This service determines if the user has the required rights to execute the given application. If this is the case, the rest of access control mechanisms take place. Otherwise, the application is not executed. The *AppLocker* uses attributes stored in the application such as signed certificate attributes, vendor name, application version, application name, etc to take the final decision. (Microsoft, 2009) (Paul Thurrott & Rafael

Rivera, 2009). It is worth noting the advantage of using attribute-based access control because it enables the policy to persist after application updates. The policies used in *AppLocker* can be either manually specified by the administrator or remotely enforced using the *Group Policy Management* tools. These tools provide a unified view of several administration tasks such as *users*, *computers*, *services*, *ACLs*, etc.

In contrast, the Linux operating system also provides different access control mechanisms. The first one is a file system access control which is available in all Linux distributions. Essentially, each securable object available in the file system (i.e., *regular file*, *directory*, *device* or *process* which are defined in Linux as special types of files) is associated with an *access right* mask which determines the permissions associated with that object. A *user* always belongs to a *group*. For this reason, there is at least a default group always available. When a user creates or executes a *process*, this process stores the *user and group identifiers* of the user for which it is acting on behalf of. Then, when the process tries to access a securable object, the access mask available for such an object is checked against the user and groups associated with the process in order to determine if it can be accessed or not. The access tuple is made up of 3 different permissions *(read, write and execute)* for three different levels of security: *user, group and others*. It is worth pointing out that this elementary access control does not originally enable the management of customized access rights for different users and groups. For this reason, this authorization model has been extended with Access Control List (ACL) in modern Linux releases. **Error! Reference source not found.** shows an architectural overview of this ACL extension.

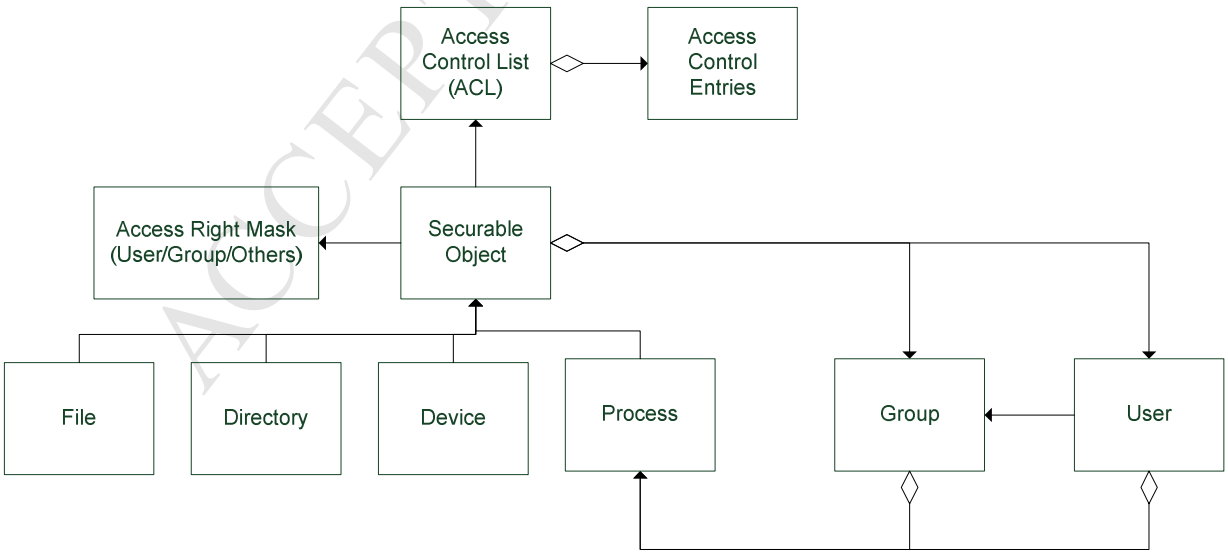

Figure 4 - Basic Discretionary Access Control and ACL implementation in Linux

Apart from the DAC mechanism, the latest Linux distributions also provide an additional access control mechanism enabled by default, which is usually provided to enable Mandatory Access Control (MAC). There are multiple Linux MAC implementations such as *AppArmor* (AppArmor), *TOMOYO* (TOMOYO), *SELinux* (Peter A. Loscocco & Stephen D. Smalley, 2001) and *GRSecurity* (Spengler, 2005) that are widely used. Some comparisons have been done between them (Michael Fox, John Giordano, Lori Stotler, & Arun Thomas) (Fox, Giordano, & Thomas), and *SELinux* has been demonstrated to be the most powerful security model (Fox, Giordano, & Thomas). This implementation is enabled by default in *Fedora Core* distributions. In *SELinux*, an authenticated *user* has assigned a set of *roles* which determine his/her assigned tasks within the organization. A role, in turn, is assigned to a set of *domains.* A domain is defined as an atomic element used to establish authorized access for the associated role. A *securable object* is associated with *security labels*. When a user tries to access a securable object, all the *subject domains* associated to such a user are matched against the *security labels* associated with the protected object. The entire security model is governed by an ACL which is composed of access entries representing security policies. This ACL is checked and only the privileges of those entry lists which match the presented information will be granted access. As a result, any potential damage (even if the source is an administrator or a root user) is prevented from affecting the entire system (Red Hat). Processes which belong to a different domain cannot interact unless a transition rule is defined.

Note that all the authorization methods described in this section for both Linux and Windows (with the exception of *AppLocker* for Windows) store the information related to the access control inside the securable file objects. This is convenient for cases in which there is at least an *admin* user predefined in the system which is a reasonable assumption for cloud computing. In this case, all the system files may belong to such a predefined user. However, this approach is not appropriate for managing files that may belong to users who are not registered already in the OS. It is worth pointing out that the authentication scheme is based on the usage of an external authentication service which may allow *unknown* users to be logged in and access particular files available therein. This calls for the design of new authorization methods and protocols in which the authorization information may be provided directly from external services that are completely managed outside the cloud. In fact, this is the approach followed by *AppLocker* which protects against both external intruders and cloud providers from executing any undesired code in the guest OS.

4. Network Security

In cloud computing, all network traffic can potentially be monitored by employing man-in-the-middle techniques. This is especially relevant for the cloud provider who can monitor the network traffic even

at the hypervisor level. Thus, strict firewall policies need to be carefully written to allow and disallow traffic of interest, with the configuration being done remotely using secure communication channels based on secure and strong key-interchange and effective encryption such as TLS v1.2.

Configuring network access control in the cloud can be a challenging task for a cloud customer requiring advanced access control to virtual infrastructures and network topology consisting of many VMs. Usually, cloud providers only offer a very limited and controlled capability for the customers who want to define networking rules between different VMs in order to reproduce a particular networking topology or configuration. As for today, advanced infrastructure deployments with well-defined firewall rules at different edges of the network do not exist. In addition, today's cloud providers do not enable the customer to manage IP address assignment. In fact, the customer usually does not even know beforehand the IP addresses used by the VMs which really hamper the firewall rule definition. Current OSs are not aware of this particular concern pertinent to cloud environments. This necessitates novel methods for defining filtering policies for the cloud. The usage of hostnames for the definition of firewall rules instead of IP addresses can be a solution but it requires the cloud provider to allow customers to manage DNS servers to resolve the hostnames inside the virtual infrastructure.

Windows firewall system has been improved considerably since the Windows Vista release when outbound firewall rules and Ethernet MAC filtering were not supported. Nowadays, Windows 7 firewall allows basic capabilities such as describing a default policy to accept/reject, filtering traffic based on IP addresses, TCP/UDP ports, Ethernet MAC address, inbound and outbound firewall rules. However, this firewall still does not support defining firewall rules based on hostnames. Such a feature can be achieved by third party software such as Zone Alarm. In Linux, the most extended framework for intercepting and manipulating network packets is Netfilter (Russell) and IPTables. The Linux IPTables utility and the Linux kernel allow the establishment of Firewall rules that can filter or transform packets. Netfilter supports basic capabilities such as the ones also supported in Windows 7. It also supports the capability to define firewall rules based on host names rather than IP addresses.

Even when using hostnames instead of IP addresses in the definition of firewall rules, the customer cannot ensure that the hostnames will be resolved against legitimate virtual machines. Thus, new techniques based on the distributed management of the DNS service between the cloud provider and the customer may be addressed together with the usage of the DNS Security Extension (*DNSSec*) which is supported by both OSs in order to protect against attacks to the DNS service. DNS is subject to DNS poisoning and spoofing attacks whereby it is possible to pollute DNS tables with malicious entries. DNSSec protocol has been invented to protect against such attacks.

Another networking security concern related to cloud environments is the definition of firewall rules based on Ethernet MAC filtering addresses. Current OSs assume a static MAC address for the networking devices but it may not be completely true for cloud computing in which network adapters are virtualized. Typically, the MAC addresses are assigned and managed by the cloud provider by means of hypervisors and are not exposed to customers. In fact, the smart management of MAC addresses may be used to support the protection of VMs traffic, live migration of VMs, VM forensics, and other advanced features in the cloud. This remains an open issue and novel methods are needed to adapt firewall rules against MAC addresses changes.

5. Accounting

Both Windows and Linux have been designed to store system logs in the local file system. In the cloud environment, it is typically stored in the VM file system. Moreover, these logs are usually saved unencrypted and can be accessed by an attacker. This represents a potential threat for the user as potentially valuable and sensitive information can be extracted from the logs. This information may include user data that is saved in the logs or knowledge about the processes performed by the VM during its operation. To protect against this threat, the OS logs should be encrypted before saving them to disk. Another option is to use a ciphered file system to store such logs where they are secured by the file system itself.

Nevertheless, having the logs stored in the local file system of the VM (either ciphered or not) may cause another potential threat regarding logs availability. In physical deployments, logs are available after the usage of the machine on the physical hard disk. However, in a highly dynamic environment like cloud computing, file systems reside in virtualized volumes and VMs are created and destroyed on demand. Thus, it is common that the cloud provider destroys the corresponding volume together with the VM when it is no longer needed in order to save resources. This implies that logs stored in the VM file system are also lost. If the VMs were compromised logs might be no longer available for future forensic analysis. A possible solution for this threat is to preserve VM volumes for some time. However, this may cause a waste of the cloud provider resources. Another approach might be to provide a separate logging system, whereby logs of VMs can be stored and secured in a separated place. This logging system can be provided and managed by the cloud provider. However, stored logs should be ciphered by the source OS that produces them in order to avoid potential attacks or information accessibility from the cloud provider. Finally, it is worth mentioning that, in order to cipher such logs, some cryptographic keys are needed. This implies some initial configuration or key provisioning into the VM, which can lead into similar problems related to the provisioning of user credentials for validation as described in Section 2.

6. Privacy and Encryption

Data privacy and encryption can be studied at different levels within a system: memory, disk and networking. Regarding memory, we still continue to struggle to develop efficient memory designs and implementations that can also support privacy of data. One of the main reasons is because of the data remanence property by which the residual representation of data remains even after the computer is shut down. Privacy of RAM memory protects the system if the memory device is physically stolen. At run-time, the memory can be protected using different approaches. Firstly, *Address Space Layout Randomization* (ASLR) (Jun Xu, 2003) is a mechanism that loads the execution code for critical applications into a random location in memory every time the system boots. Both Linux 2.6.38 and Windows 7 provide this kind of protection. The locations of the heap, stack, Process Environment Block, and Thread Environment Block are also randomized. Such a technique makes it more difficult to extract sensible data such as keys from memory dumps which are produced at run-time, but it is still not impossible to do so. Additionally, memory can be also protected by disabling any FireWire port. This kind of port enables direct access without security restrictions to the memory, providing a way to perform a memory dump of the system (Martín, 2007).

Regarding disk encryption, both OSs provide complete disk encryption capabilities. Windows 7 natively provides the *BitLocker* (Ferguson., 2006), while Linux 2.6.38 provides *dm-crypt/LUKS* (Fedora). These types of software are able to encrypt the complete disk (i.e. all disk partitions), including system partitions, swap partitions/files, and hibernate files. Both of them place a bootable software in the Master Boot Record (MBR), which is in charge of requesting the decryption key in order to gain access to the disk to be able to load the OS. They support two-factor authentication: for instance by using a Universal Serial Bus (USB) device or a TPM Trusted Platform Module (TPM module together with a key manually inserted by the user). Moreover, they use passphrase strengthening in order to provide protection against dictionary attacks. The main difference between them is the operation modes which are supported by both applications. BitLocker only supports the Cipher Block Chaining (CBC) mode where initialization vectors are statically derived from the sector number. *Dm-crypt* not only supports this operation mode, but also others such as the IEEE P1619 standardization project for encryption of stored data XTS-AEX (Committee, 2007). Both operation modes are based on the Advanced Encryption Standard (AES) symmetric encryption algorithm and they provide complete protection for the encrypted data especially when VMs are not running.

File encryption is another additional privacy feature supported in both operating systems. Linux also relies on the *dm-crypt* software for the encryption of single files, whereas Windows 7 uses the *Encryption File System* (*EFS*) component (Bragg, 2012). In essence, a file is encrypted using a

symmetric key generated. Then, for each user which is authorized to access the file, a copy of the key is encrypted with the user's public key and it is stored in the file's metadata.

Regarding network encryption, the latest DirectAccess (Microsoft) feature of Windows 7 provides an easy and secure way of establishing Virtual Private Networks (VPNs). It allows the automatic establishment of a bi-directional connection from client computers to the corporate network. It uses Internet Protocol version 6 (IPv6), and Internet Protocol security (IPsec) to authenticate both the computer and the user. Thus, it establishes two different IPSec tunnels. The first one is an IPSec Encapsulating Security Payload (ESP) tunnel using the computer certificate that enables authentication to be requested on behalf of the user. Then it establishes a second IPSec ESP tunnel that authenticates the user and provides access to intranet resources. If a native IPv6 network is not available at the client side, it uses 6to4 or Teredo (Huitema, 2007) to send IPv4-encapsulated IPv6 traffic. Moreover, if the client computer cannot use 6to4 or Teredo to reach the *DirectAccess* server (e.g., because of a firewall or proxy), the client automatically connects by using IP over Hypertext Transfer Protocol Secure (HTTPS). Unlike most VPN connections, *DirectAccess* can separate intranet traffic from Internet traffic, thus reducing unnecessary client traffic on the VPN corporate network.

There are also different open source tools for establishing VPNs in Linux. Openspawn (Openswan) is a popular implementation of the IPSec protocols for Linux to establish secure VPNs. This implementation provides its own IPSec kernel stack and it can also use the code included in recent kernels. Regarding VPNs over Secure Sockets Layer (SSL), it is worth mentioning OpenVPN (Feilner, 2006) because it is the most used and it is pretty mature. Although all the capabilities provided by the Windows 7 DirectAccess tool can be addressed by the VPN solutions available for Linux, it seems that the configuration and management of DirectAccess tool of Windows is more intuitive and easy to use than setting up VPNs in Linux.

One of the most important open issues in cloud computing nowadays is data privacy. Both OSs provide efficient volume encryption capabilities which are strong enough to keep data safe even when they are stored on a third-party device. The usage of a strong security mechanism based on the combination of a TPM/USB device and a user-inserted key in order to generate the encryption/decryption key provides significant privacy and protection of the data. Today, data encrypted either with *BitLocker* or *DM-Crypt* has been proven as secure as when they are stored in a VM which is shutdown. This prevents the cloud provider from gaining access to the file system when VMs are powered off.

However, the VMs continue to be vulnerable when they are running. The file system decryption key is located at some point in the memory and the VM is susceptible to memory dumps at run-time stage by

someone trying to have access to such valuable information. Memory dumps of the complete system require administration privileges. Thus, it is really important to disallow privileged users in an OS running into the cloud. In case that a customer needs to perform any administrative task, he/she can perform it on the OS image running locally at the client side and the image can then be uploaded to the cloud. In addition, OS utilities have to be tweaked to avoid the execution of non-authorized applications and to avoid direct accesses to the memory without security checks (e.g., by means of the *FireWire* channel). However, a memory dump can be directly taken by using any snapshot capability provided by the virtualization software running in the physical host which is managed by the cloud provider. This memory dump can be later analyzed to find sensitive information and keys.

Recently, all tools using AES-based encryption are being rapidly cracked by mean of the extraction of the AES keys when memory dumps are available. This fact questions the suitability of AES for cloud computing environments in which memory dumps can be easily done by the cloud provider. To better understand the reason, a deep knowledge of AES algorithm is required. We briefly review this algorithm which is based on the famous Rijndael AES encryption algorithm (Daemen & Rijmen, 2001) and uses several iterative rounds of encryption each of which consists of four main different steps (as shown in Figure 5).

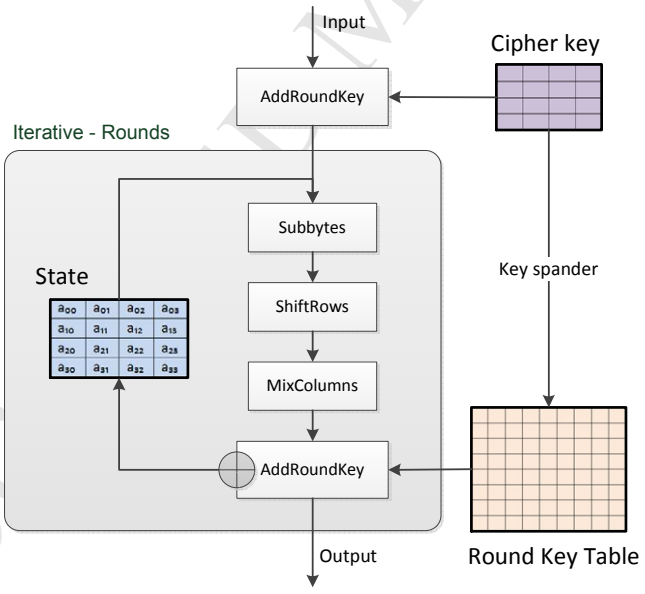

Figure 5. Block Diagram of the Rijndael AES encryption algorithm

The *addroundkey* step combines the current state table with one of the keys derived from the original cipher key. It uses one different derived key obtained from the round key table at each round of the algorithm. The derived keys are calculated using a key schedule which expands the original short key into several separate round keys. This way of generating keys imposes an important vulnerability in

memory dumps. In a given time during the AES execution, there will be a part of the memory dump in which a linear sequence of bytes (the expanded key) can be predicted by applying such a direct mathematical transformation over the previous bits (the short key). This fact significantly reduces the time required to calculate possible AES keys available in the memory dump to a few minutes. This is one of the main reasons why today there is no an efficient data encryption algorithm for the cloud that can protect volume images even against memory dumps and remains a clear open issue that still needs to be addressed to develop an efficient data encryption solution for cloud computing. Assuming an algorithm is able to fulfill such security requirements, security attacks against the volume image may not decrypt the information in acceptable times.

Apart from volume encryption, all the application-level services deployed in VMs which need to use sensible information may be designed using just-in-time encryption procedures in which sensible information is only decrypted during the minimal required time, carrying out the operations in the associated services, and finalizing with a re-encryption procedure. This just-in-time decryption can be achieved using the current file encryption capabilities available in both OSs.

7. Discussion

Our analysis presented in the previous sections show that the data security and privacy are the two main concerns of cloud computing nowadays, resulting in a slow migration and adoption of cloud services. As discussed, existing cloud solutions for addressing security and privacy clearly have weaknesses and shortcomings. This calls for designing new cloud computing security solutions in which the architecture does not assume a trusted cloud provider which may require access to the guest OS images in order to properly configure them for use in a cloud environment. This assumption may not be acceptable for many organizations planning to migrate to the cloud.

A critical feature to incorporate in cloud computing design is to provide cloud customers with the ability to upload and use their own guest OS images which can be managed, prepared, and controlled by themselves. By taking advantage of such functionality, customers have increased control and we can reduce scenarios where the customer cannot decide what to run in the cloud. The number of cloud providers providing this kind of functionally is scarce nowadays. To date, uploading such guest OS images in an encrypted fashion is not being provided by cloud providers at all. The security analysis presented in previous sections has shown that it is essential to encrypt the guest OS image from the very beginning even before uploading the guest OS to the cloud in order to ensure that neither the end users nor the cloud provider can have access to the OS when the VMs are powered off. The solutions identified in this work for authentication and authorization systems for cloud computing require that

the cloud provider should not have access to the guest OS image; otherwise, the cloud provider may potentially alter the configuration setting of the OS in order to use non-legitimate security services and information. This indeed calls for novel design to deal with encrypted volumes in cloud computing environments which in turn may cover the complete life-cycle of such volumes including uploading, executing, mounting, deleting, etc. This design feature is more related to the functionality provided by the cloud computing platform than to the features of the guest OS running into the VMs. However, the issue is completely different to protect access to the VMs. As discussed, it is important to develop novel efficient cryptographic methods to embed within the OSs which can protect against memory dumps. It has been demonstrated that the current cryptographic algorithms do not resist this kind of threat. This is clearly one of the primary root causes for which current commodity operating systems do not fit in the virtualized environments of cloud computing.

An important security feature to be yet supported in cloud computing is memory encryption at run-time. This entails the use of Address Space Layout Randomization (ASLR) not only in executable processes but also for data stored in memory and random indices of memory page table. This table can be dumped and can be used to restore the linear address space of the data. This is a clear vulnerability from which RAM data can be easily leaked from the cloud provider using the hypervisor. Thus, a strong memory encryption feature is required to protect against memory dumps. Several efforts have recently proposed various solutions (such as CryptoPages (Duc & Keryell, 2006) and more recently CryptKeeper (Peterson, 2010) to address memory encryption at run-time and the first implementations for Linux are already available, e.g., METAL (Levy & Khan, 2007). It is worth pointing out that applying an efficient memory encryption process in the OS fixes the problem directly related to the volume encryption because keys available in memory are also encrypted and protected against memory dumps.

In today's cloud computing environment, the cloud provider has the ability to resume a snapshot of a user's VM, recreating the original state of the snapshot which is of a major security concern. Moreover, in a cloud environment, all user sessions in the VM are done by means of remote connections to the VM. These connections are done with tools such as Secure Shell (SSH), Remote Desktop Connection (RDC), Virtual Network Computing (VNC), etc. As a result, the cloud provider can always see the logon interface when a snapshot is resumed.

8. Conclusion

This paper has presented a security review of Windows 7 and Linux Fedora Core 15 with kernel 2.6.38 which are two versions of modern Windows-based and Linux-based Operating Systems. A survey of various security aspects such as authentication, authorization, security protocols, network

security, memory, privacy and cryptography has been carefully examined. Security features of these two operating systems have been analyzed when deployed in a cloud computing environment. Our analysis has shown that today's security features provided by these OSs have clear security concerns and weaknesses. We have also highlighted possible future solutions to address those weaknesses that we have identified. Neither current Windows nor Linux operating systems are very secure when being hosted on VMs running in cloud computing environments. Moreover, the paper discussed key challenges and open issues that still need to be addressed to implement more secure OSs for the cloud.

References

A. Melnikov and K. Zeilenga. (2006). *Simple Authentication and Security Layer (SASL).* RFC.

Abella, S. A. (2006). Obtenido de Data Execution Prevention. Changes to Functionality in Microsoft Windows XP Service Pack 2, Part 3: Memory Protection Technologies: http://www.microsoft.com/technet/

AppArmor. (2010). *AppArmor.* Obtenido de http://wiki.apparmor.net/index.php/Documentation

Bar-Lev, A. (2010). *OpenSC - tools and libraries for smart cards.* Obtenido de http://www.opensc-project.org/

Bragg, R. (2012). *The Encrypting File System.* Obtenido de TECHNET: http://technet.microsoft.com/en-gb/library/cc700811.aspx

Committee, I. C. (2007). Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices.

Conover, M. (April de 2010). *Analysis of the Windows Vista Security Model.*

Daemen, J., & Rijmen, V. (2001). *The design of Rijndael: AES--the advanced encryption standard.* Springer.

Duc, G., & Keryell, R. (2006). CryptoPage: An Efficient Secure Architecture with Memory Encryption, Integrity and Information Leakage Protection. *Annual Computer Security Applications Conference ACSAC06.* Ieee.

Fedora. (2011). *Fedora 14 Installation Guide.* Fultus Corporation.

Feilner, M. (2006). *OpenVPN: Building and Integrating Virtual Private Networks.* Packt Publishing.

Ferguson., N. (2006). *AES-CBC + Elephant diffuser : A Disk Encryption Algorithm for Windows Vista.*

Fox, M., Giordano, J. S., & Thomas, A. (2008). *SELinux and grsecurity: A Side-by-Side Comparison of Mandatory Access Control and Access Control List Implementations.*

Huitema, C. (2007). *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs).* RFC, Microsoft.

Jun Xu, Z. K. (2003). Transparent runtime randomization for security. *22nd Sympomsium on Reliable Distributed Systems,* (págs. 260-269).

Leach, P. (April de 2003). *Security Service Provider Interface.* Microsoft Developer Network (MSDN), Microsoft. Obtenido de Security Service Provider Interface.

Lettice, J. (2004). *Windows v Linux security: the real facts.* Obtenido de http://www.theregister.co.uk/2004/10/22/linux_v_windows_security/

Levy, J., & Khan, B. (2007). Hiding Your Wares: Transparently Retrofitting Memory Confidentiality into Legacy. *IEEE International Conference on Communications 2007.*

Martín, A. (2007). *FireWire Memory Dump of a Windows XP Computer: A Forensic Approach.*

Michael Fox, John Giordano, Lori Stotler, & Arun Thomas. (2008). *SELinux and grsecurity: A Case Study Comparing Linux Security.* University of Virginia.

Microsoft. (28 de 2003). *Logon and Authentication Technologies.* Obtenido de TECHNET: http://technet.microsoft.com/en-us/library/cc780455(WS.10).aspx

Microsoft. (March de 2009). *Windows 7 Security Enhancements.* Obtenido de TECHNET: http://technet.microsoft.com/en-us/library/dd548337%28WS.10%29.aspx

Microsoft. (2011). *Microsoft NTLM.* Microsoft Developer Network (MSDN), Microsoft.

Microsoft. (2012). *http://technet.microsoft.com/en-us/network/dd420463.aspx.* Obtenido de TECHNET: http://technet.microsoft.com/en-us/network/dd420463.aspx

Molnar, I. (2009). *Exec-shield*. Obtenido de Exec-Shield: http://people.redhat.com/mingo/exec-shield/

Neuman, C. Y. (July de 2005). Obtenido de The Kerberos Network Authentication Service (V5): http://www.ietf.org/ rfc/rfc4120

Openswan. (2012). *Openswan.* Obtenido de http://www.openswan.org

Paul Thurrott, & Rafael Rivera. (2009). *Windows 7 secrets – Do what you never thought possible with Windows.* John Wiley & Sons Ltd.

Peter A. Loscocco, & Stephen D. Smalley. (2001). Meeting Critical Security Objectives with Security-Enhanced Linux. *Linux Symposium.* Ottawa.

Peterson, P. (2010). Cryptkeeper: Improving security with encrypted RAM. *IEEE International Conference on Technologies for Homeland Security (HST).* IEEE.

Red Hat. (2010). *Security Enhanced Linux User guide.* Obtenido de http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/Security-Enhanced_Linux/index.html

Riley, S. (2006). *Mandatory integrity control in Windows Vista.* Obtenido de http://blogs.technet.com/b/steriley/archive/2006/07/21/442870.aspx

Russell, R. (2002). *Packet Filtering HOWTO.* Obtenido de http://iptables.org/documentation/HOWTO//packet-filtering-HOWTO.html

Samar, V. (1996). Unified login with pluggable authentication modules (PAM). *Proceedings of the 3rd ACM conference on Computer and communications security.* New Delhi, India: ACM.

Spengler, B. (2005). *Increasing performance and granularity in role based.*

TOMOYO. (2012). *TOMOYO.* Obtenido de http://tomoyo.sourceforge.jp/

K. Salah is an associate professor of Computer Science. He received the B.S. degree in Computer Engineering with a minor in Computer Science from Iowa State University, USA, in 1990, the M.S. degree in Computer Systems Engineering from Illinois Institute of Technology, USA, in 1994, and the Ph.D. degree in Computer Science from the same institution in 2000. He has over ten years of industrial experience in embedded systems and software and firmware development of network protocol stacks and device drivers for ATM and Ethernet. Dr. Salah is currently with the Computer Engineering Department, Khalifa University of Science, Technology and Research (KUSTAR). His primary research interests are in the area of operating systems, computer networks, network security, cloud computing, performance evaluation and modeling.

Sherali Zeadally received his Bachelor's Degree in Computer Science from University of Cambridge, England, and the Doctoral Degree in Computer Science from University of Buckingham, England in 1996. He is an Associate Professor at the University of the District of Columbia. He currently serves on the Editorial Boards of over 15 peer-reviewed international journals. He has been serving as a Co-Guest editor for over a dozen special issues of international scholarly journals in the areas of networking. He is a Fellow of the British Computer Society and a Fellow of the Institution of Engineering Technology, UK. His research interests include computer networks including wired and wireless networks), network and system security, mobile computing, cloud computing, RFID, performance evaluation of systems and networks.

JOSE M. ALCARAZ CALERO (jmalcaraz@um.es) holds a Ph.D. in computer Science by University of Murcia. Currently, he is a lecturer at University of Murcia, Spain and research staff at Hewlett-Packard Laboratories, United Kingdom. He has published several articles in journals and conferences including IEEE and IET Transactions. He is associate editor of a numbering of reputable international journals such as IET Wireless Sensor Systems and Journal of Advances in Network and Communications. His research areas include cloud computing, virtualization, security, operating systems, policy-based systems and semantic web.