In Proceedings of the The 1st International Workshop on Dependability of Clouds, Data Centers and Virtual Computing Environments (DCDV, with DSN'11), Hong Kong, June 2011

Lucy in the Sky *without* Diamonds: Stealing Confidential Data in the Cloud

Francisco Rocha*, Miguel Correia†
*Universidade de Lisboa, Faculdade de Ciências and Carnegie Mellon University – Portugal/USA
†Instituto Superior Técnico / INESC-ID – Portugal
rocha.francisco@gmail.com, miguel.p.correia@ist.utl.pt

***Abstract*—Cloud Computing is a recent paradigm that is creating high expectations about benefits such as the pay-per-use model and elasticity of resources. However, with this optimism come also concerns about security. In a public cloud, the user's data storage and processing is no longer done inside its premises, but in data centers owned and administrated by the cloud provider. This may be a concern for organizations that deal with critical data, such as medical records.**

We show that a malicious insider can steal confidential data of the cloud user, so the user is mostly left with trusting the cloud provider. The paper achieves this goal by showing a set of attacks that demonstrate how a malicious insider can easily obtain passwords, cryptographic keys, files and other confidential data. Additionally, the paper shows that recent research results that might be useful to protect data in the cloud, are still not enough to deal with the problem. The paper is a call to arms for research in the topic.

I. Introduction

Cloud Computing is seen by many as the next big paradigm shift in information technology. The myriad of new cloud commercial solutions indicates that the cloud is being widely accepted. The benefits of moving to the cloud have already been discussed in multiple texts, e.g., [3]. The question that still needs to be answered is whether or not a secure transition to the cloud is possible for organizations that require data confidentiality.

Cloud security is a topic with many facets [7], but the paper focus only on security from the perspective of the *user* of a *public cloud* [15], a company, governmental organization or other. More specifically, the paper is about confidentiality of the user's data, a concern of many potential users. The reason of alarm has its root in the deep changes that the cloud paradigm brings to information technology. The most relevant change is that the user relinquishes control of its data to the cloud provider, which stores and processes it in its own data centers. The user may hesitate to trust the cloud provider and/or the foggy legal situation of data in the cloud.

The paper considers the problem of the *malicious insider* in a cloud based on the *Infrastructure as a Service* (IaaS) model [15]. A IaaS cloud offers the user computing resources, i.e., a set of virtual machines in which the user can run arbitrary software on top of an operating system of its choice. The classical mechanism to protect the confidentiality of data is to encrypt it. However, in an IaaS cloud, this procedure is not enough because data needs to be arbitrarily manipulated in the virtual machines, which are in the provider's data center. These infrastructures are managed by administrators that have at least remote access to the servers in which the virtual machines run (see for instance open source cloud infrastructures like Open Stack [1] or Eucalyptus [17]). If an administrator goes rogue, an example of the well-known security threat of a malicious insider [13], he can leverage the entry in these machines to gain access to the user's data. This conspicuous problem lead us to analyze the possible attack vectors such an attacker can exploit and if it is currently possible to prevent it.

Cloud providers are aware of the malicious insider threat and argue that they have solutions to mitigate the problem [10]. A first solution is not to allow physical access to the servers. However, all attacks presented in this paper are done remotely, they do not require physical access. A second mechanism is a zero tolerance policy for insiders that access data stored in the cloud. This measure only takes place after the attack has happened, which is not enough, for instance, in the case of an attacker that was fired or left voluntarily. The third mechanism consists in logging all accesses to the servers where the users' data is stored. These logs are later used for internal audits, to find out if employees are behaving according to privacy policies. Again, this solution only detects the attack after it happened, which may be too late. All these mechanisms are important and should be deployed, but they are not enough to prevent any of the attacks in the paper.

The objective of the paper is to show that a malicious insider can steal confidential data of the cloud user, therefore, currently, the user is mostly left with trusting the cloud provider. The paper achieves this goal in two steps. The first step, and main contribution of the paper, is the presentation of four attacks showing that a malicious insider can easily compromise passwords, cryptographic keys, files, and other confidential data. Second, the paper discusses how a set of recent research mechanisms fail to protect data from the previous attacks. This does not mean that these mechanisms are not useful, they are, but only that they do not solve the

problem we consider here. In a nutshell, the paper is a call to arms to more research in the topic.

II. Virtualization and IaaS Cloud Computing

The foundation of the IaaS cloud service model is *native* virtualization [18]. In native virtualization, a *hypervisor* or *virtual machine monitor* provides an abstraction of the hardware resources to a set of *virtual machines* (VM) executing on top of it (see Figure 1). A VM includes its own operating system that runs above the hypervisor.

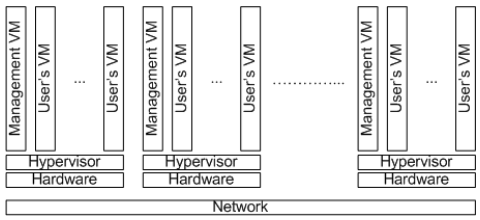

Figure 1. Abstract architecture of an IaaS service, with servers running an hypervisor and users' virtual machines

As already mentioned, the paper focus on the IaaS cloud service model, as provided by *Amazon EC2* or Open Stack / Eucalyptus. In this service model, the user runs its systems in a set of VMs in a set of physical servers. Each server has a *management VM* that is responsible for the administration of the VMs of the cloud users (on the left of the servers in the figure). The attacks were executed using the Xen hypervisor, in which the management VM is denominated *dom 0* [4].

III. Attacking Confidentiality in the Cloud

This section presents four attacks that a malicious insider can perform to access the user's data. These attacks clearly demonstrate that it is currently possible to violate the confidentiality of the cloud user's data. The attacks are available as short videos in YouTube (links in the description of each attack).

The test environment was a single machine, with an Intel Core 2 vPro processor and 4GB RAM. The machine had a Trusted Platform Module (TPM) version 1.2, a module that provides certain security services [21]. The cloud infrastructure was emulated by the Xen 4.0.1-rc4 hypervisor. The management VM was dom 0 running Ubuntu Server 10.04 with Xen-enabled kernel version 2.6.32.15. There was a single user VM running Ubuntu Server 10.04 and Apache version 2. The Apache web server had a private key used for authentication and establishing secure channels with the clients (browsers).

In the attacks we assume the following basic model. The attacks are performed by a malicious insider that has root access to the management VM of the servers that compose the cloud, i.e., to dom 0 in our environment. We also assume that, on the contrary, the attacker has no credentials that allow him to authenticate himself on the user's VM.

A. Cleartext passwords in memory snapshots

The first attack is quite simple. It takes advantage from how easily an administrator can obtain a memory dump, or snapshot, of any VM. To obtain the memory dump he needs to issue a single command while logged as root in the dom 0. It has already been shown that it is possible to extract passwords in clear text from a Linux memory dump [8]. We demonstrate that this is also possible against cloud user VMs.

To perform the attack, a malicious insider obtains a memory dump from the target VM using the *dump-core* command that is part of the Xen management user interface (*xm*). The *dump-core* command performs a dump of the memory region reserved to the targeted VM. The attacker only needs to identify the VM in the command. In our case, to confirm the presence of passwords in the dump file, we used the *cat* command to send the dump into the standard output that was redirected to the *strings* and *grep* commands in order to locate the known password.

These two steps are illustrated in Figure 2. The passwords found were those used for a user to login and by Apache to allow using an RSA key (suggestively named *loginpwd* and *apachersapwd* in the figure). A video that shows the attack is available online (http://www.youtube.com/watch?v=FJryJ3gYSkc).

In a real attack the malicious insider has no previous knowledge about the password. In this scenario, the attacker simply needs to devise a method to take advantage of the information contained in the dump file filtered by the *strings* command. It has already been demonstrated that for some applications, e.g., TrueCrypt, it is even possible to automate the search for passwords once the attacker obtains a memory dump. However, the important here is to show that the passwords are available to the attacker.

```
$ xm dump-core 2 -L lucidomu.dump
Dumping core of domain: 2 ...
$ cat lucidomu.dump | strings | grep loginpwd
loginpwd
loginpwd
$ cat lucidomu.dump | strings | grep apachersapwd
apachersapwd
apachersapwd
apachersapwd
```

Figure 2. Attack that finds cleartext passwords in the memory of a VM

B. Obtaining private keys using memory snapshots

The objective of the second attack is to obtain the private key of a private-public key pair. In the example of this attack that we demonstrate, the key is the one used by the Apache server to establish secure channels with its clients. With this key, the attacker can impersonate the server before its clients.

Apache keeps the private key in memory, so it can be found in a memory dump, just like the passwords in the previous attack. However, the key is only a number, e.g.,

1024 or 2048 bits if RSA is used. Apparently, this fact provides a sort of security by obscurity, because all the memory is a sequence of numbers with an unclear semantics at first sight. This attack shows that this obscurity provides no security.

To execute the attack, the malicious insider first obtains a memory dump of the target VM, like in the previous section. After this first step, the attacker has the keys, but they are hidden in at least hundreds of megabytes. We used the same technique to find keys that obtained good results in the cold boot attack [11]. Cryptographic keys are typically stored in memory in a format that is recognizable. The most used format is PKCS#1 [19] that represents keys as an ASN.1 object. Such objects have a structure in memory that is known. It includes the sequence identifier (the byte 0x30) followed by several bytes and the 0x02 01 00 02 pattern. There is a tool denominated *rsakeyfind* (part of a package with the same name available for several Linux distributions) that performs this type of search, so the attacker can use it on the dump file and extract the keys. This search mechanism can lead to false positives, but we did not obtain any while using the tool. The attack is illustrated in Figure 3 and in the video online (http://www.youtube.com/watch?v=8xKhS5ZGR5s).

```
$ xm dump-core 2 -L lucidomu.dump
Dumping core of domain: 2 ...
$ rsakeyfind lucidomu.dump
found private key at 1b061de8
version =
00
modulus =
00 d0 66 f8 9d e2 be 4a 2b 6d be 9f de 46 db 5a
...
publicExponent =
01 00 01
privateExponent =
...
prime1 =
...
prime2 =
...
```

Figure 3. Attack that finds RSA private keys in the memory of a VM

C. Extracting confidential data from the hard disk

The motivation for this third attack is an hypothetical scenario in which a cloud user is aware of the dangers of memory snapshots, so it only accepts to use a service in which the ability to create them is disabled. The attack leverages the fact that the user's data has to be stored in disk, which is controlled by the management VM, or dom 0 in the Xen case.

In the specific attack that we demonstrate, we consider that Linux's *Logical Volume Manager* (LVM) is used. LVM creates logical volumes on top of the physical storage in a machine. LVM uses three levels of abstraction. The first handles the physical storage resources, called *physical volumes*. The second handles *volume groups*, which are sets of physical volumes. The third level handles *logical volumes*. It is at this level that we can mount or create file systems.

The process to extract confidential information from disk volumes is similar to performing a backup copy of the disk partition owned by the victim VM. To obtain the backup copy, the malicious insider runs the following sequence of steps in the dom 0. First, the attacker creates a snapshot of the victim VM drive using the LVM's *lvcreate* command, which creates a new logical volume. In the second step, the attacker uses the *kpartx* utility to add the partition mappings from the partition table existent on the newly created snapshot volume. The *kpartx* utility, derived from the *partx* command, is responsible for reading partition tables existent on devices and to create the respective device maps. In the third step, the attacker performs a scan with LVM's *vgscan* command to find all available LVM physical volumes and volume groups. The objective of the malicious insider is to locate the volume groups that belong to the victim VM. After finding the desired volume group, the attacker activates the logical volumes existing in it. To do this he can use the *vgchange* command with the *-ay* option.

With the logical volumes activated, the attacker can mount them at will and extract the desired information using arbitrary commands. For example, he can simply use the *rsync* file copying utility to copy as many files as he desires to another machine. To get the system back to its initial state, the attacker starts by un-mounting and deactivating the logical volumes with the *vgchange* command. Then, he removes the partition mappings with the *kpartx* command and removes the snapshot logical volume using the *lvremove* command.

This sequence of steps is depicted in Figure 4 and the video of the attack is online (http://www.youtube.com/watch?v=wvlnIA49spY).

```
$ lvcreate -L 2G -s -n lv_st /dev/main_vol/domu
Logical volume 'lv_st' created
$ kpartx -av /dev/main_vol/lv_st
...
$ vgscan
Found volume group 'LucidDomU'
$ vgchange -ay LucidDomU
$ mount /dev/LucidDomU/root /mnt/
$ rsync -avhp /mnt/ /media/backup
...
$ umount /mnt/
$ vgchange -an LucidDomU
$ kpartx -d /dev/main_vol/lv_st
$ lvremove /dev/main_vol/lv_st
```

Figure 4. Backup a VM's LVM logical volume

D. Virtual machine relocation

The fourth and last attack considers a hypothetical cloud with a security level much higher than what we considered so far. This cloud uses integrity-protected hypervisors, i.e., hypervisors that have a certain configuration and are protected against modification [23], [20]. This protection is

obtained by using a trusted boot process that involves the Trusted Platform Module (TPM) to assure the user that a certain hypervisor and software configuration was started. This allows the cloud to prove that the hypervisor is a special version in which, e.g., snapshots of memory and accesses to the disk area were disabled.

Despite these protections, relocating users' VMs in different servers is a basic functionality of IaaS clouds that we believe cannot be disabled in practice. The attack consists in using this functionality to escape from the protections we just mentioned, then executing some of the previous attacks. This attack was first presented in [20], so our contribution is only to demonstrate it.

This attack can be divided in two main phases. In the first, the objective of the malicious insider is to assure the user that the server where it is about to launch its VM is using an integrity-protected hypervisor, just like the cloud would normally do. The attacker will do this by using *remote attestation*. This attestation is possible due to the TPM hardware module present in the servers. In our case, we are going to demonstrate the attack using a direct proof mechanism based on the PrivacyCA [2]. We designate rather informally by *secure server* a machine with an integrity-protected hypervisor and by *malicious server* a machine controlled by the attacker and that does not have an integrity-protected hypervisor. Please use Figures 5 and 6 as companions to understand the steps of the attack.

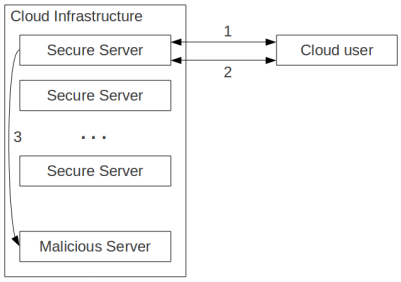

Figure 5. Virtual machine relocation attack

To perform the attestation, the attacker first needs to obtain the *endorsement key certificate* from the TPM of the machine the user (challenger) wants to verify, i.e., of the secure server. This is achieved using the *getcert* command. This certificate is part of the certificate chain to the root *certification authority* (CA), which is passed to the *aikpublish* application[1] that generates an *attestation identity key pair* (AIK). The private key of the pair is used to sign PCRs quotations sent to the user (details below). This application also generates a proof file that can later be used by the user to verify that the AIK is valid.

In Figure 5, arrow 1 represents the stage when the proof file is used by the user to authenticate the secure server.

[1]The *aik** commands are provided by the PrivacyCA [2].

```
PCR-02: ED DF ...
PCR-04: 59 91 ...
PCR-05: 97 99 ...
PCR-07: BA C3 ...
SecureServer$ ./getcert ek_cert.crt
SecureServer$ ./aikpublish ek_cert.crt inter_ca_03.crt
  root_ca.crt proof_file_provider.out aik_blob.key
SecureServer$ ./aikrespond aik_blob.key
  encrypted_challenge_client.out decrypted.out
SecureServer$ cat decrypted
Francisco Rocha
SecureServer$ ./aikquote aik_blob.key 2 4 5 7 pcrs_quote.out

CloudUser$ cat secret.in
Francisco Rocha
CloudUser$ ./aikchallenge secret.in
  proof_file_provider.out encrypted_challenge.out
  aik_rsa_pub.key
CloudUser$ ./aikqverify aik_rsa_pub.key pcrs_quote.out
  2 eddf...
  4 5991...
  5 9799...
  7 bac3...

SecureServer$ ssh admin@MaliciousServer

MaliciousServer$ xm create lucidLAMP.cfg
MaliciousServer$ xm list
Name                          ...
Domain-0                  ...
LucidLAMP                  ...
```

Figure 6. Attack that relocates a VM

The process consists in the user running the *aikchallenge* application to create an encrypted challenge using the public AIK and sending it in the proof file to the secure server. Then, the secure server uses the *aikrespond* application to respond to the challenge, using the private AIK to decrypt it and prove to the user that it is the owner of that public-private AIK key pair.

After the authentication, the secure server uses the *aikquote* application to create a quote of some of its TPM *platform configuration registers* (PCRs), in the case of Figure 6, PCRs 2 4 5 and 7, and sends it to the cloud user. The PCRs are set during the boot process with hashes of parts of the software, namely of the hypervisor and dom 0. Upon reception of the PCRs quote, the user uses the *aikqverify* application with the public AIK to verify that it came from the correct machine and that the configuration is the one that should be. The hash that corresponds to the hypervisor must match the one that the user knows to correspond to the integrity-protected hypervisor in which it trusts. The verification process is represented by arrow 2 in Figure 5.

The second phase of the attack is the following. After the attestation process is complete, the verifier trusts the hypervisor. Then, the attacker relocates the VM about to be launched into a different server: the malicious server. This step is illustrated by arrow 3 in Figure 5 and by the last three commands in Figure 6. This machine would be running an insecure version of the hypervisor and dom 0, offering resources to the attacker that can help him compromising confidential data existent in the VMs, for example, the

snapshot command of the Xen management user interface discussed in previous sections.

We provide an illustration of the attack in video (http://www.youtube.com/watch?v=Z9o9-pAtSp0).

IV. Protection Mechanisms and Their Limitations

This section discusses how recent research mechanisms fail to protect the confidentiality of users' data from the previous attacks. Notice that the authors of these mechanisms do not claim to solve the problem; we selected these mechanisms because they seemed close enough to our problem. Therefore, we also do not mean to say that these mechanisms are not useful – they are – only that they do not solve the problem we consider in the paper. Some of these protection mechanisms can be seen as a solution trying to solve the whole confidentiality problem inherent to cloud computing [20], [6], whereas others are only mechanisms that could be used as a part of a more complete solution [5].

This section also points research directions that may be pursued in order to solve the problem studied in the paper. If cryptography seems problematic due to the negative result of [22] (next section), trusted computing and distributing trust seem to be promising directions (Sections IV-B and IV-C).

A. Cryptography

Cryptography may be seen at first sight as the solution for data confidentiality in the cloud. However, data in IaaS clouds has to be manipulated by the applications that run in the user VMs, which normally can not happen if the data is encrypted. For instance, a payroll processing application cannot process payrolls if all data is encrypted. There has been some discussion about fully homomorphic encryption (FHE) as a solution for this problem, because FHE allows certain operations, like addition, to be executed over encrypted data [9]. However, currently the performance of FHE makes this infeasible. Furthermore, van Dijk and Juels have shown that even with FHE, cryptography can not enforce privacy in the cloud if data from several clients is processed [22].

B. Trusted Computing

Trusted computing is a promising technology to solve the novel security challenges existent in cloud computing. Having the TPM as hardware-based root of trust may enable researchers to devise strong security solutions. This subsection is about relevant work published in this area.

Virtual TPM (vTPM): The first mechanism is the vTPM [5], which provides a set of virtual TPMs to be used in virtualized environments, so that every VM can use its own private TPM. The objective is for an operating system (OS) running on a VM to have the same use model and TPM command set as an OS that runs directly over the hardware platform and takes advantage of the physical TPM.

The vTPM approach is not a complete solution for user data confidentiality in the cloud. Having vTPMs for VMs does not guarantee that the hypervisor is not vulnerable or that the VMs are launched or migrated to a trustworthy server. Furthermore, security issues in this technology have been exposed [16]. The problem is that the lack of communication between the *libxc* domain builder and the vTPM software makes the vTPM vulnerable to a time-of-check-time-of-use (TOCTOU) attack. A second problem is the unpredictable size of the *trusted computing base* (TCB), i.e., of the software that is critical in terms of security. The size is unpredictable because with new VMs the system will need to create the corresponding vTPM, so the size of the TCB grows, which is problematic from the security viewpoint [14].

PVI: The Private Virtual Infrastructure (PVI) is an attempt to solve the security challenges of the cloud [12]. The main idea is to have the cloud user control a PVI, while the cloud provider controls the cloud fabric. The PVI architecture uses the vTPM and the Locator Bot (LoBot) protocol as building blocks. The vTPM provides the root of trust while the LoBot protocol allows each virtual machine to be remotely verified by its owner. Although the objective of the PVI is to provide security for the cloud user, it has no specific mechanisms to protect the confidentiality of the user's data.

TCCP: The most relevant solution for the problem considered in the paper is the Trusted Cloud Computing Platform (TCCP) [20]. The main purpose of the TCCP is to provide a closed box environment for guest VMs running in the cloud, i.e., to prevent administrators from inspecting or tampering with the contents of the users' VMs. The TCCP also offers remote attestation to a cloud user wishing to launch a VM in an untrusted cloud provider.

The trusted computing base (TCB) of the TCCP includes two main components: the trusted virtual machine monitor (TVMM) and the trusted coordinator (TC). The latter is managed by a trusted third party, outside the cloud and independent of the cloud provider. The TCCP operates by creating a set of trusted nodes that run a TVMM (akin to an integrity-protected hypervisor). The TVMMs are used to run the user's VMs and are managed by the TC. The TCCP aims at protecting both the VM bootstrap and migration operations, which are critical stages of VM management.

Although this solution is on the right track, the addition of a TC maintained by an external trusted entity is problematic: it requires trust in the third party and may be unacceptable for the cloud provider.

C. Distributed trust

A different kind of solution consists in distributing trust among several cloud providers. In this approach the user

does not need to trust the cloud provider and its administrators. Instead, it trusts that there is no collusion among malicious insiders of more than a certain number of clouds, which is a weaker assumption.

Currently this idea has been applied in the context of cloud computing by a single system, DepSky [6]. The purpose of DepSky is to guarantee confidentiality, integrity and availability of data stored in the cloud. DepSky, however, does not provide the IaaS model, but only a storage service similar, e.g., to Amazon S3. DepSky scatters data in several storage clouds of different cloud providers, e.g., in four. It uses Byzantine quorum system algorithms to assure the integrity and availability of data, even in the presence of failures data losses or corruptions in some of the clouds. The confidentiality of the data is assured by encrypting it and storing the encryption key in all the clouds using secret sharing. This solution allows the implementation of secure storage clouds, but does not protect data in IaaS clouds that provide VMs to the user. Therefore, it does not prevent the attacks that we demonstrate in the paper.

V. Conclusion

Cloud Computing is a promising paradigm with growing acceptance, but there is still much work to be done if we want to achieve holistic security in the cloud. The paper focused on the malicious insider threat. It demonstrated that this type of attacker can violate data confidentiality without the need of high technical skills. It also discussed the benefits and limitations of current partial solutions for the problem.

Acknowledgments

This work was partially supported by the European Union FP7/2007-2013 under project TCLOUDS (grant agreement 257243), and by Fundação para a Ciência e a Tecnologia through the RC-Clouds (PCT/EIA-EIA/115211/2009) and the Multiannual and CMU-Portugal Programmes. The title of the paper is partially taken from a song by Lennon/McCartney.

References

[1] Open Stack. http://www.openstack.org.

[2] PrivacyCA. http://www.privacyca.com.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, April 2010.

[4] P. Barham, B. Dragovic, K. Fraiser, S. Hand, T. Harris, A. Ho, R. Neugebaurer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 164–177, October 2003.

[5] S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: Virtualizing the trusted platform module. In *Proceedings of the 15th USENIX Security Symposium*, pages 305–320, August 2006.

[6] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa. DepSky: Dependable and secure storage in a cloud-of-clouds. In *Proceedings of the European Conference on Computer Systems (EuroSys)*, pages 31–46, April 2011.

[7] Cloud Security Alliance. Top threats to cloud computing v1.0, March 2010.

[8] S. Davidoff. Cleartext passwords in Linux memory. http://philosecurity.org/pubs/davidoff-clearmem-linux.pdf, July 2008.

[9] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 169–178, 2009.

[10] E. Grosse, J. Howie, J. Ransome, J. Reavis, and S. Schmidt. Cloud computing roundtable. *IEEE Security Privacy*, 8(6):17 –23, 2010.

[11] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *Proceedings of the 17th USENIX Security Symposium*, pages 45–60, August 2008.

[12] F. J. Krautheim. Private virtual infrastructure for cloud computing. In *Proceedings of the Workshop on Hot Topics in Cloud Computing*, June 2009.

[13] P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell. The inevitability of failure: The flawed assumption of security in modern computing environments. In *Proceedings of the 21st National Information Systems Security Conference*, pages 303–314, 1998.

[14] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig. TrustVisor: Efficient TCB reduction and attestation. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 143–158, May 2010.

[15] P. Mell and T. Grance. The NIST definition of cloud computing, July 2009.

[16] D. G. Murray, G. Milos, and S. Hand. Improving Xen security through disaggregation. In *Proceedings of the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 151–160, 2008.

[17] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus open-source cloud-computing system. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid*, pages 124–131, 2009.

[18] M. Rosenblum and T. Garfinkel. Virtual machine monitors: Current technology and future trends. *IEEE Computer*, 38(5):39–47, May 2005.

[19] RSA Laboratories. PKCS #1 v2.1: RSA cryptography standard. ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf, June 2002.

[20] N. Santos, K.P. Gummadi, and R. Rodrigues. Towards trusted cloud computing. In *Proceedings of the Workshop on Hot Topics in Cloud Computing*, June 2009.

[21] Trusted Computing Group. Trusted computing platform module main specification. version 1.2, revision 94. http://www.trustedcomputinggroup.org, 2006.

[22] M. van Dijk and A. Juels. On the impossibility of cryptography alone for privacy-preserving cloud computing. In *Proceedings of the 5th USENIX Conference on Hot Topics in Security*, pages 1–8, 2010.

[23] A. Vasudevan, J. M. McCune, N. Qu, L. van Doorn, and A. Perrig. Requirements for an integrity-protected hypervisor on the x86 hardware virtualized architecture. In *Proceedings of the 3rd International Conference on Trust and Trustworthy Computing*, pages 141–165, June 2010.