International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Trusted Framework for Secured Information Storage over Cloud Environment

P. Pothuraju[1], Dasari Rajesh[2]

[1]Computer Science and Engineering, Rise Gandhi Group of Institutions, Ongole, India

Abstract: ***Cloud orchestration involves cloud resources scaling up and down, management, as well as manipulation to better respond user's requests and to facilitate operational objectives of the service providers. These promote the elastic nature of cloud platform but force upon significant challenges to cloud service providers. Particularly, security issues such as inconsistency may arise while dynamic changes such as virtual machine migration occur. In this paper, we intend to tackle this problem, specifically for intrusion detection/ prevention and VPN/IPsec as main security mechanisms. More precisely, we propose a systematic verification approach to check the compliance of security configurations. To this end, we first elaborate on two properties, namely intrusion monitoring configuration preservation and VPN/IPsec protection configuration preservation. Then, we derive a set of formulas that compare security configurations before and after migration. This allows reasoning on whether the aforementioned security properties hold. To this end, we encode these formulas as constraint satisfaction problems.***

Keywords: Cloud security, security architecture, security and privacy.

1. Introduction

Recent developments in virtualization have made cloud computing an increasingly important research and operational area. Migration of Virtual Machines (VMs) has opened new opportunities in computing. It can help in many ways such as high-availability of services, consolidated management, and workload balancing. While virtualization and VM migration provide important benefits, their combination may introduce new security challenges. In order to ensure a secure cloud, firewalls should not be the only line of defense. Moreover, access to and from the VMs have to be tracked and monitored for eventual security attacks. At the same time, the number of VMs deployed in a data center may increase enormously. Therefore, cloud providers have to offer intrusion monitoring solutions, which should be adaptive and scalable for the elastic cloud environment. In this regard, traditional hardware security appliances are fundamental to secure cloud infrastructures. However, the traffic that is between collocated VMs normally remains at the virtual level and passes through virtual switches, which makes the hardware appliance blind to this type of traffic. To be effective in a virtualized environment, security controls must live inside the virtual and cloud systems. While traditional hardwarebased solutions are not able to respond to virtual machine activities, virtual security solutions can be adaptive, scalable, and capable of addressing these challenges. Therefore, a virtualized intrusion detection and prevention system is also required to monitor traffic on the virtual level. To this end, various approaches are being proposed by research initiatives [1] and standard bodies [2] to inspect the VMto-VM traffic.

In addition to Intrusion Detection and Prevention Systems (IDPS), tenants can ask for the deployment of a secure Virtual Private Network (VPN) between their corporation networks and their networked VMs running in the cloud. This is to enable protecting information in transit over insecure networks or leveraging the cloud services as an extension of their corporate data centers. The necessity of implementing a VPN connection between home network or the corporation data center, and the VMs deployed in the public cloud has been realized by virtualization leaders and cloud providers. For instance, IPsec VPN connection is supported in Amazon [3]. In addition, VMware propose vShield Edge VPNs [4] for cloud providers. Elasticity of cloud computing implies addition, mobility, or even removal of VMs and consequently, requires reconfiguration of network nodes, including security appliances. Some research initiatives have proposed solutions to address the implementation of dynamic reconfiguration in the cloud [5], [6]. However, dynamic reconfiguration is error-prone [7], and if not properly performed may cause security configurations inconsistencies, thus exposing the VMs as well as the whole infrastructure to serious security threats. Therefore, a security policy verification framework at the cloud management layer is essential to enable cloud providers certifying that the right security policy is enforced at the destination location. In this paper, we propose such a framework within a scope of intrusion detection and IPsec. It is based on the comparison of a given known secure configuration, existing before migration, with a newly generated configuration, that should be deployed at the destination. The main goals are to detect errors in the configuration and to provide a useful feedback to correct the problem before the actual migration takes place. The verification spans both source and destination data centers in the case of a migration across data centers. The main contributions of this paper are threefold:

- Define the concepts of intrusion monitoring and IPsec protection preservations in cloud data centers.
- Derive a set of formulas relating: (1) the traffic monitored by the IDPS devices before and after migration (2) the traffic protected via IPsec endpoints before and after migration. For the sake of completeness, the verification is performed over the traffic in source and destination data centers for all hosted VMs.

Licensed Under Creative Commons Attribution CC BY

- Elaborate a verification approach based on Constraint Satisfaction Problem (CSP), for establishing security preservation.

2. Related Work

In this section, we present a literature review with respect to intrusion monitoring, and IPsec VPN in cloud computing, with focus on the analysis of security policy consistency, and the verification of security policy compliance.

2.1 Intrusion Monitoring

With respect to IDS, a number of initiatives focus mainly on proposing a solution to handle inspection specifically designed to the cloud. For instance, [1] proposes virtual machines introspection (VMI). This approach relies on the hypervisor or the virtual machine monitor to inspect the VM from the outside of the host. Modi et al. [8] present a survey on various intrusion detection techniques in the cloud. Roschke et al. [9] and Dhage et al. [10] propose management architecture for distributed IDS. Azmandian et al. [11] present a hypervisor-based approach for detecting intrusive activities. In [12], security issues related to the virtualization technology are reviewed and a comparison between traditional and modern monitoring techniques is presented along with the weaknesses as well as their protection and assurance levels. Dhage et al. [10] propose an IDS architecture to be deployed in a distributed cloud computing environment where separate instances are installed for each user and managed by a single controller. In [13], an IDS as a Service is proposed. The latter is a network and signature-based IDS for the cloud that monitors and logs network activities between virtual machines within a pre-defined Amazon virtual private cloud.

On the other hand, works such as [14], [15] propose model checking to analyze the correctness of IDS configurations. More precisely, Tekaya et al. [14] use model checking of temporal logic formulas to verify the correctness of the intrusion detection system using the SMV model checker. The properties are either verified if the behavior is normal, or violated if the behavior is intrusive. In [15] a model checking verification approach is presented to detect specification errors in attack signatures of intrusion detection.

The attack signatures specified using the Event Description Language (EDL) are transformed to PROMELA and are verified using the model-checker SPIN. Uribe et al. [16] propose an approach for modeling and reasoning about the configurations of a combination of network intrusion detection systems (NIDS) and firewalls based on constraint logic programming. Song et al. [17] propose a formal framework for the analysis of intrusion detection systems that use declarative rules for attack recognition. This is to prove that a given IDS can detect all attacks that would violate the security requirements of a given system. Stakhanova et al. [18] present a framework for the analysis of host and network-based IDS for conflict detection in the rule-sets. In [19], a framework based on Event Calculus (EC) for formal analysis of intrusion detection systems is presented. It checks that security requirements are preserved at runtime by monitoring the satisfaction of the corresponding EC formulas.

2.2 IPsec VPN

As far as IPsec policy analysis and conflict detection are concerned, to the best of our knowledge, there is very little research work done towards these objectives. There is a belief that the VPN tunnels are statically created when needed by the actors and therefore there could not be any erroneous IPSec configurations. But this is not the case in an elastic cloud environment where VMs are moved around. Most of the existing works focus on detecting and/or resolving anomalies in a single IPsec device (called intra-policy conflicts) and/or between multiple IPsec devices (called inter-policy conflicts). As a first initiative, Fu et al. [20] examine such anomalies and propose an algorithmic approach to detect and resolve such anomalies. The algorithm is meant to verify that a given IPsec implementation satisfies a set of pre-defined requirements. Hamad et al. [7] model IPsec policy using Ordered Binary Decision Diagrams (OBDDs) and propose an algorithmic approach based on Boolean operations over OBDDs to detect anomalies. In [21] Niksefat and Sabaei improve the efficiency of the approach in [7] by eliminating the need of processing all the rules. Khoury et al. [22] propose hierarchical colored Petri nets for specifying network data traffic and abstract functions for modelling IPsec mechanisms operations. Compared to these works, the present paper tackles a different problem, which is verifying that the modification of IPsec configuration after migration has been consistently performed. Furthermore, we use the well-established SAT-solving based technique, which have been demonstrated to be more effective than BDD-based approaches [23]. Jarraya et al. [24], [23] have examined security policy verification in the context of VM migration, but only consider verification of stateless firewall configuration in the cloud environment. The present work aims at verifying other security mechanisms, namely intrusion detection and VPN/IPsec, which configuration settings and objectives are different from firewalls.

3. Encoding Ids and IpSec Configurations as Constraints

Constraint satisfaction is the process of finding a solution to a propositional reasoning problem specified using a vector of variables that must satisfy a set of constraints. A solution is therefore a vector of values that satisfies all constraints. Many problems including those of scheduling, test generation, and verification can be encoded as CSP. Constraint satisfaction problems are typically identified with problems based on constraints on a finite domain. More formally, a CSP is defined by a set of variables $\{x_i\}_{1\leq i\leq n}$ and a set of constraints $\{C_j\}_{1\leq j\leq m}$. Each variable x_i is defined within a domain D_i of possible values. Each constraint C_j involves all or a subset of the variables and specifies the allowable combinations of values for that variables. A statement of the problem is defined by an assignment of values to some or all of the variables. A consistent or legal assignment is one that does not violate any constraint. A complete assignment is one in which all variables are

Licensed Under Creative Commons Attribution CC BY

assigned values. Programs that solve CSP problems are called constraint solvers. We use the Sugar CSP-solver, a SAT-based constraint solver based on a new SAT-encoding method, namely order encoding [25]. Sugar accepts Lisp-like expressions. For instance, the constraint $C_1 \wedge C_2$ is equivalent to the expression (and C_1 C_2) in Sugar syntax. The complete language accepted by Sugar can be found in [26]. There are two possible outputs for a problem submitted to Sugar: either satisfiable (denoted hereafter as SAT), if all constraints are satisfied or unsatisfiable (denoted hereafter as UNSAT), otherwise. For instance, for a conjunction of constraints $c_1 \wedge \cdots \wedge c_2$, a SAT conclusion allows to infer that $\{c_i\}_{1 \leq i \leq n}$ are not disjoint whereas UNSAT conclusion asserts the contrary.

In this work, we consider a Snort-based configuration format. Snort is an open source widely deployed IDS. A Snort rule consists of two sections, a rule header and rule options. The rule header contains criteria for matching a rule against data packets, and the action to be taken in case a packet matches. The options part usually contains an alert message as well as information about the parts of the packet that should be used to generate the alert message. The options part may also contain additional criteria for matching a rule against data inside the packets. There are three major action directives supported by Snort: pass, log, or alert. Pass simply drops the packet. Log action writes the full packet to the logging routine. Alert action generates a notification event using the user-specified method, and then logs the packet for later analysis. In the following, we present how we encode IDS configurations in Sugar.

The CSP variables are the set of integer variables needed to encode the monitoring attributes of IDS rules. In order to represent an IP address, 4 integer variables within the range [0, 255] are used. A source (resp. destination) IP address is represented by $\{sip_i\}_{1 \leq i \leq 4}$ (resp. $\{dip_i\}_{1 \leq i \leq 4}$). The integer variable pr $\in$[0, 255] represents the protocol number. We also define two other integer variables to encode the source and destination port numbers, respectively ps and pd within the range of values [0, 65535]. We encode the direction (ingress, egress or bidirectional) using an integer variable dr $\in$ [0, 1] so that 1 represents bidirectional traffic and 0 represents unidirectional traffic such that we switch the IP addresses and the ports from source to destination and vice versa to represent ingress or egress traffic. The action is encoded using a variable act $\in$ [0, 2] so that 0 represents pass, 1 represents log, and 2 represents alert. To encode the options we use a variable opt $\in$ [0, 40000] so that each value represents a unique option value. Since it is possible to have more that one option, we encode them as a logical conjunction formula of all options. The action in the rule header is invoked only when all criteria in the options are true. Thus, to encode IDS rules in CSP, we use a set of integer variables {act, pr, sip1, sip2, sip3, sip4, ps, dr, dip1, dip2, dip3, dip4, pd, opt}. Each single IDS rule predicate p is encoded as a CSP constraint. The latter is a conjunctive logical formula over the variables in V with their corresponding values specified in the IDS rule. More precisely, a CSP constraint is written as act = $v_1 \wedge$ pr = $v_2 \wedge$ sip1 = v_3 ... $\wedge$ dip4 =$v_{12} \wedge$ pd = $v_{13} \wedge$ opt = v_{14} where v_i is to be replaced by the actual value in the corresponding IDS rule. The IDS is then encoded as a constraint C built as a disjunctive logical formula over all constraints of the IDS rules. Thus, the list of IDS rules $\{R_n, R_{n-1}, \ldots\ldots, R_1\}$ are encoded as the logical formula $R_1 \vee R_2 \vee \cdots \vee R_n$.

In order to encode IPsec configuration in Sugar, we reuse a subset of the aforementioned CSP variables namely {pr, sip1, sip2, sip3, sip4, ps, dip1, dip2, dip3, dip4, pd, act}. The IPsec protocols ESP and AH are encoded using variable pr (pr = 50 for ESP and pr = 51 for AH). To encode the mode (transport or tunnel), we define a variable md, which values are in {0, 1} such that 0 encodes transport mode and 1 encodes tunnel mode. For action, we use a variable act, which has values in [0, 2] such that 0 encodes discard, 1 encodes bypass, and 2 encodes protect. For the case of tunnel mode, the destination gateway is encoded using a variable gw and its values are in [1,m] such that m is the maximum number of gateways in the network. Also, a variable param is defined to encode the authentication or cryptographic algorithms being used such as 3DES, MD5, and so on. Its values are in [1, n] such that n is the number of authentication and cryptographic parameters available in the configuration. In the case of an empty GW configuration, the corresponding constraint will be the truth value „false".

4. Conclusion

In this paper, we proposed a verification approach that checks the consistency of security configurations related to IDPS and VPN/IPsec services. To this end, we defined new security properties, namely intrusion monitoring and VPN/IPsec protection preservations, and we derived a set of formulas, which verification allows concluding on whether these properties hold. These formulas are then encoded in constraint satisfaction problems to be solved using the SAT-based constraint solver Sugar. Our approach enables cloud providers to automatically verify that at each migration occurrence, security level of the hosted VMs (including the migrating VM) is preserved. It also helps in detecting and correcting the configuration errors if the verified properties are violated. For future work, we consider implementing our approach and elaborating on the possible deployment architectures. Therefore, we plan to develop a software tool that can automatically perform the verification with minor human interactions.

References

[1] T. Garfinkel and M. Rosenblum,"A Virtual Machine Introspection Based Architecture for Intrusion Detection," in In Proc. Network and Distributed Systems Security Symposium, 2003, pp. 191–206.
[2] C. S. Alliance, "SecaaS Implementation Guidance, Category 6: Intrusion Management," September 2012.
[3] Amazon. (2013) Amazon Virtual Private Cloud FAQs. [Online]. Available: http://aws.amazon.com/vpc/faqs/#C5
[4] VMware, "Securing hybrid clouds with vmware vshield edge vpns: A guide for providers of vcloud powered services," http://www.vmware.com/files/pdf/vmware-vshieldtechnical- brief.pdf, 2012, accessed May 2013.
[5] S. Shin and G. Gu, "CloudWatcher: Network Security Monitoring using OpenFlow in Dynamic Cloud

Licensed Under Creative Commons Attribution CC BY

Networks (or: How to Provide Security Monitoring as a Service in Clouds?)," in Network Protocols (ICNP), 2012 20th IEEE International Conference on. IEEE, 2012, pp. 1–6.

[6] T. Wood, K. K. Ramakrishnan, J. Van Der Merwe, and P. Shenoy, "Cloudnet: A Platform for Optimized WAN Migration of Virtual Machines," University of Massachusetts, Tech. Rep., 2010.

[7] H. Hamed, E. Al-Shaer, and W. Marrero, "Modeling and Verification of IPSec and VPN Security Policies," in the Proc. Of the 13th IEEE International Conference on Network Protocols (ICNP), Nov. 2005.

[8] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A Survey of Intrusion Detection Techniques in Cloud," Journal of Network and Computer Applications, vol. 36, no. 1, pp. 42 – 57, 2013.

[9] S. Roschke, F. Cheng, and C. Meinel, "Intrusion detection in the cloud," in Dependable, Autonomic and Secure Computing, 2009. DASC '09. Eighth IEEE International Conference on, Dec. 2009, pp. 729 –734.

[10] S. N. Dhage and B. B. Meshram, "Intrusion Detection System in Cloud Computing Environment," International Journal of Cloud Computing, vol. 1, no. 2, pp. 261–282, 2012.

[11] F. Azmandian, M. Moffie, M. Alshawabkeh, J. Dy, J. Aslam, and D. Kaeli, "Virtual Machine Monitor-based Lightweight Intrusion Detection," SIGOPS Oper. Syst. Rev., vol. 45, no. 2, pp. 38–53, July 2011.

[12] F. Tsifountidis, "Virtualization Security: Virtual Machine Monitoring and Introspection," Master's thesis, Royal Holloway, University of London, UK, 2010.

[13] T. Alharkan and P. Martin, "IDSaaS: Intrusion Detection Systemas a Service in Public Clouds," in CCGRID, 2012, pp. 686– 687.

[14] I. B. Tekaya, M. Graiet, and B. Ayeb, "Intrusion Detection with Symbolic Model Verifier," in ICSEA 2011, The Sixth International Conference on Software Engineering Advances, 2011, pp. 183–189.

[15] S. Schmerl, M. Vogel, and H. K¨onig, "Using Model Checking to Identify Errors in Intrusion Detection Signatures,"Int. J. Softw. Tools Technol. Transf., vol. 13, no. 1, pp. 89–106, January 2011.

[16] T. E. Uribe and S. Cheung, "Automatic Analysis of Firewall and Network Intrusion Detection System Configurations," in Proceedings of the 2004 ACM workshop on Formal methods in security engineering, ser. FMSE '04. New York, NY, USA: ACM, 2004, pp. 66–74.

[17] T. Song,"Formal reasoning about intrusion detection systems," Ph.D. dissertation, University of California, Davis, 2007.

[18] N. Stakhanova, Y. Li, and A. A. Ghorbani, "Classification andDiscovery of Rule Misconfigurations in Intrusion Detection and Response Devices," in Proceedings of the World Congress onPrivacy, Security, Trust and the Management of e-Business, ser. CONGRESS. Washington, DC, USA: IEEE Computer Society, 2009, pp. 29–37.

[19] M. Rouached, H. Sallay, O. B. Fredj, A. Ammar, K. Al-Shalfan, and M. Ben, "Formal Analysis of Intrusion Detection Systems for High Speed Networks," in Proceedings of the 9th international conference on Advances in e-activities, information security and privacy. Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2010, pp. 109–114.

[20] Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, and C. Xu, "IPSec/VPN Security Policy: Correctness, Conflict Detection and Resolution," in the Proc. of the 2nd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY), ser. LNCS, vol. 1995. Springer, 2001, pp. 39–56.

[21] S. Niksefat and M. Sabaei, "Efficient Algorithms for Dynamic Detection and Resolution of IPSec/VPN Security Policy Conflicts," in Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, ser. AINA '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 737–744.

[22] H. E. Khoury, R. Laborde, F. Barr`ere, M. Chamoun, and A. Benzekr, "A Formal Data Flow-Oriented Model For Distributed Network Security Conflicts Detection," in Network Operations and Management Symposium, 2008. NOMS 2008. IEEE, March 2012, pp. 20–27.

[23] Y. Jarraya, A. Eghtesadi, M. Debbabi, Y. Zhang, and M. Pourzandi, "Formal Verification of Security Preservation for Migrating Virtual Machines in the Cloud," in SSS, 2012, pp. 111–125.

[24] "Cloud calculus: Security Verification in Elastic Cloud Computing Platform," in CTS, 2012, pp. 447–454.

[25] N. Tamura and M. Banbara, "Sugar: A CSP to SAT TranslatorBased on Order Encoding," in the Proceedings of the SecondInternational CSP Solver Competition, 2008, pp. 65–69.

[26] "Syntax of sugar," http://bach.istc.kobe-u.ac.jp/sugar/current/docs/syntax.html, last Accessed: March 2013.

Licensed Under Creative Commons Attribution CC BY