Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 78 (2016) 33 – 39

Procedia Computer Science

International Conference on Information Security & Privacy (ICISP2015), 11-12 December 2015, Nagpur, INDIA

Placement of Security Devices in Cloud Data Centre Network: Analysis and Implementation

Santosh Kumar Majhi[a,b,*], Sunil Kumar Dhal[b]

[a]*Veer Surendra Sai University of Technology, Odisha-768018, India*
[b]*Sri Sri University, Odisha, India*

Abstract

Due to extensive use of Cloud services and newly evolving security threats, most cloud service providers (CSP) deploy varieties of security devices such as, firewalls, IPSec, IDS, etc. for controlling resource accesses based on the data centre security requirements. Today CSPs are looking for systematically hardening the security by incorporating multiple security devices in the network in a cost-effective way. In this paper, we present an automated framework for synthesizing data centre security configurations. We take a dummy data centre topology, CSP security (connectivity and isolation) requirements and business constraints (usability and cost) as input; and then synthesizes the correct and optimal data centre security. It determines the optimal placement of different security devices in the data centre.

Peer-review under responsibility of organizing committee of the ICISP2015

Keywords: Cloud data centre; Security devices; Security configuration; Cloud service provider

1. Introduction

Cloud data centre security consists of the provisions and policies adopted by an administrator to prevent and monitor unauthorized access, misuse, modification, or denial of a computer network and network-accessible resources. Now

* Corresponding author. Tel.: +91-9438403651
*E-mail address:*santoshism9@gmail.com

Peer-review under responsibility of organizing committee of the ICISP2015
doi:10.1016/j.procs.2016.02.007

days the service providers are more concerned with their security along with their products. These are becoming more fine-grained due to extensive use of various cloud services and newly evolving security threats. In addition to this, most of the organizations not only emphasizing on enforcement of the security constraints but also requiring satisfaction of different business constraints like, cost, usability demand, etc. The implementation of strong defence in a cloud data centre exploring different security design alternatives as well as resolving the contention between the security and business constraints is an important but challenging problem.

Usually, the cloud service provider security requirements cover: (i) connectivity requirement that defines the service flows between various data centre devices; and (ii) partition or separation requirement that defines various partition patterns as combination of different security devices (firewall, IPSec, IDS, and NAT etc.) and their relative arrangements based on the security enabling device capability. A partition pattern signifies the level of security resistance, for example, traffic filtering (firewall), IPSec based encryption, payload traffic inspection (IDS), hiding traffic source identity (NAT) and/or combination of these constraints.

On the other hand, the service provider business constraints include service usage and cost of service. Installation of different security enabling devices significantly affects these constraints. For example, implementing both IPSec and IDS instead of firewall might cause some usable application inaccessible from a server, thereby reducing the service usability. The major requirement is to find usable data centre security configuration by exploring various security design alternatives (partition levels) that increases the service usability without significant degradation of the overall data centre security. At the same time, it is required to find best security partition in affordable cost. Therefore, a careful balancing between the security and business constraints is required to determine correct and cost-effective data centre security configurations.

In our work we consider only the placement of security devices between servers is defined using four partition patterns - Firewall, IPSec, IDS and NAT – to meet the security needs in cost effective manner. In other words, optimized placement of devices with minimum cost requirements is described here. Now, let us understand the partition patterns and working of them (firewall, IPSec, IDS, and NAT).

2. Partition

Data centre administrators can define partition patterns considering different combination of security devices (primitive partition s). They can objectively exclude some of the device combinations according to requirements or domain knowledge of Cloud data centre. Administrators can also define the capability of different partition patterns by providing relative arrangement of these patterns. Now, we formalize the network partition as a set of rules, $\{ir_1,\ldots,ir_n\}$, where, each partition rule ir_i is formally represented as follows:

$$ir_k: (src = i) \wedge (dst = j) \wedge (service = x) \Rightarrow D_{ij}^{k}(x)$$

The decision variable, $D_{ij}^{k}(x) = 1$ indicates that the corresponding k^{th} partition pattern is required to be deployed between node pair $(i; j)$ under service x. Here, k represents the relative arrangement order of the partition pattern. A possible set of primitive partition patterns with their relative order is presented in Table 1. It shows that $k = 1$ for "firewall Deny" and $k = 3$ for "IPSec based authentication", etc. So, if $D_{ij}^{k}(x) = 1$, then the service x must be denied between the node pair (i, j) through firewall.

Table 1: Partition Patterns

Partition Order	Partition Pattern	Decision Variable	Symbol Used
1	Firewall Deny	$D_{ij}^{1}(x)$	
2	IPSec Encryption	$D_{ij}^{2}(x)$	
3	Payload Inspection (IDS)	$D_{ij}^{3}(x)$	
4	Source Identity Hiding (NAT)	$D_{ij}^{4}(x)$	

Other symbols used for client is and  router is 

3. Placement of Security Devices

Here, we consider a dummy data centre topology, organizational security (connectivity and partition) requirements and business constraints (usability and cost) as input; and then synthesizes the correct and optimal security configurations that maximizes partition while satisfying the requirements.

Here the algorithm generates a correct and cost-effective network security configuration (partition patterns between each pair of nodes) using constraint satisfaction checking. However, the optimal placement of security devices in the network may not be determined based on the partition results. This is because of the fact that there may have similar types of partition (say, firewall) between a sources *i* and multiple destinations *j*, which signifies to deploy multiple firewalls between all such node pairs. This placement might not be optimal as in that case, a single firewall might be sufficient to place at the source *i*. Therefore, given partition patterns for each pair of nodes in the network, it is required to find minimum number of devices for enforcing the security constraints. We need to present a procedural approach for determining optimal placement of the security devices.

Partition Equivalence Cluster Creation: This process logically combines all nodes, *j* with similar partition configuration with respect to a specific node *i* into a single network cluster.

Node Partition Equivalence Cluster :An *Partition Equivalence Cluster* , $C_i^k$is defined as a logical collection of servers which have same partition value with respect to a node *i* under partition pattern *k*.

An partition cluster with respect to a node *i* under partition pattern *k* is represented as follows:

$$C_i^k = U_{j\epsilon N} \{j | D_{ij}^k = 1\}$$

We procedurally create the partition equivalence cluster with respect to each node *i* in the network.
Now, multiple partition clusters may contain overlapping member nodes. Thus, these clusters should be further combined to group the nodes optimally in terms of partition s. This is done by creating different partition partite classes based on the similarity in cluster members.
Partition Partite Class: An *Partition Partite Class,* S_{ix}^{k} is defined as logical collection of partition clusters C_x^k w.r.t a cluster C_i^k such that $C_x^{k+} \subseteq C_i^{k+}$ and the node *x* does not belong to C_i^k. Here, in C_x^{k+}, the plus sign indicates that the clusters cannot be empty if they are going to take part in the formation of the partition partite class.
The partite class S_{ix}^{k} is formally represented as follows:

$$S_{ix}^{k} = C_i^k \cup_x \{(C_x^k \mid C_x^k) \subseteq C_i^k \wedge \neg(x \in C_i^k)\}$$

This process signifies reducing the redundancies between the partition clusters. Each *partition class* has two parties; the suffix sequence (here, $i \in N$ and $x \in N$) of the class, S_{ix}^{k} the left partite(nodes that will belong in the left group of the partition device); and the participating clusters of class S_{ix}^{k} represent right partite.
It is to be noted that, if the node, *x* belongs to the corresponding *determine* partition cluster, C_i^k, then x is not included in the class, S_{ix}^{k}.

This is because of the fact that there may exist an partition (of same pattern *k*) between the node *x* and the other member nodes of the cluster, C_i^k. After creating different partite classes, we procedurally determine the security device placement between different clusters under the partite classes. The pseudo code of this process is presented in **Algorithm 2.**

Before the implementation of algorithm we need to know the input and output patterns. Input pattern is of .csv extension files, i.e. comma separated values and the output file is of .txt extension, i.e. text file. The output file is in the format which is read by graphviz application and a corresponding graph according to the nodes, clusters, classes and partition patterns is drawn for better visualization.

In input file, i.e. .csv file we enter the cluster number and the number of nodes in it with varying partition patterns. For example, suppose we have two clusters C_1 and C_2 having 3,4,5 and 4,5 as their corresponding nodes between which partition patterns 2,3 and 1,2 are required respectively. So, in .csv file we write them as 1,2,3,0,3,4,5,-1 and 2,4,5,0,4,5,-1. Here, 0 indicates that the following digits are meant to be read as nodes and -1 is the termination of a single line (i.e. details of a cluster is complete). At the end of file we put a $ sign to indicate the end of file.

Algorithm 1: Class creation

1. Select initial cluster and count its number of nodes in it.
2. Check the other clusters, except the initial one, which have less or equal ‘number of nodes’ as the initial one.
3. Once step 2 is satisfied, check if the ‘nodes’ in initial one matches exactly with the next corresponding clusters.
4. If step 3 is satisfied check for their similar partition patterns and those which can be overlapped or discarded and then combine the clusters together in the same class.
5. Continue this till all clusters are taken into consideration.
6. Create next class and goto step 1.

Algorithm 2: Deriving Security Device Placement

1. Begin
2. Create a sorted list $S'(s_1, s_2, \ldots, s_N)$ based on descending order of $|S_{ix}^{k}|$ such that $s_1 \equiv S_{x1x2}$.
3. Select first element s_1 from S'.

4. Place a k^{th} level security device between the left partite $\{x1x2..\}$ and the right partite, clusters in S_{x1x2}. In this the k^{th} level security can be anything ranging from single partition pattern to multiple partition pattern. To simplify k^x, x can be anything like 1,2,3,4 or 1234 or combinations of these four partition patterns which is described in detail in the upcoming examples.
5. Remove all the clusters $\{C_{x1}, Z_{x1},..\}$ from $S'(s_1,s_2,\ldots,s_N)$
6. Remove all the *empty sets* from S'.
7. Goto Step 2 if S' is *non-empty*.
8. End.

We first create a sorted list S′ of partite classes based on decreasing order of class size, $|S_x^k|$, the total number of distinct nodes in all participating clusters in a class. Then, we procedurally select each element class, s_i (each element represents a partite class, Sx1x2..) in sequence from S′ and place a security device between the two partites of that class. After the corresponding device placement, we remove from all
the classes, the clusters that are already considered. This process is applied continually until the list S′ becomes an empty set. In this way, for each partition pattern k, our framework derives the optimal device placement in the network. Let us see few examples for the above procedure.

Example 1

Let us implement an example for single partition pattern where *k=1*. Consider, the following partition equivalent clusters under k^{th}partition pattern (determined based on the partition variables) for a network of 6 nodes, (1,2, 3, 4, 5, and 6): C_1^k = {2,3,4}; C_2^k = {3,4}; C_3^k = {4,5}; C_4^k = {2,3}; C_5^k = {2,3}; C_6^k = {3}

The associated partition classes for these clusters are as follows: S_{156}^k= {C_1, C_5, C_6}; S_{26}^k= {C_2, C_6}; S_3^k= {C_3}; S_{456}^k={C_4, C_5, C_6}; S_{546}^k= {C_5, C_4, C_6}; S_6^k= {C_6}.

Now, the different iterations of the device placement procedure are shown as follows:

Iteration1: S′(S_{156}, S_{456} S_{546}, S_{26}, S_3, S_6); (1, 5,6) Δ{C_1 C_5 C_6} ~(2, 3, 4)

Iteration2: S′(S_2= {C_2}; S_3= {C_3}; S_4= {C_4}; S_5={C_4}); (2) ΔC_2~(3, 4)

Iteration3: S_3= {C_3}; S_4= {C_4}; S_5= {C_4}); (3) ΔC_3~(4, 5)

Iteration4: S_4= {C_4}; S_5= {C_4}); (4) ΔC_4~(2,3)

Here, x Δy denotes that the security device need to be placed between the nodes/node groups x and y.

Following is the output of the above example from the graphviz application. It shows all the connectivity and security device placement of the required pattern.

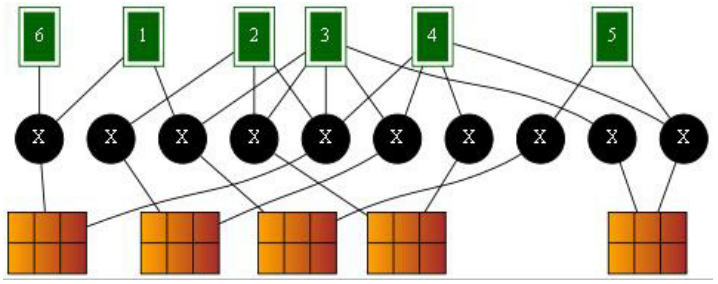

Fig. 1. showing the detailed connectivity and partition pattern of example 1.

Example 2

Let us implement another example for single partition pattern where *k=2 i.e.* IPSec. Consider, the following partition equivalent clusters under k^{th}partition pattern (determined based on the partition variables) for a network of 4nodes, (1,2, 3 *and* 4):

C_1^k = {3, 4}; C_2^k = {4}; C_3^k = {1}; C_4^k = {2, 3};

The associated partition classes for these clusters are as follows:
S^k_{11} = {C_1, C_2}; S^k_2= {C_2}; S^k_3= {C_3}; S^k_4= {C_4};
Now, the different iterations of the device placement procedure are shown as follows:
Iteration1: S′(S_{12}, S_4, S_3, S_2); (1, 2) ∆{C_1,C_2 } ~(3, 4)
Iteration2: S′(S_4, S_3); (4) ∆{C_4 } ~(2,3); Iteration3: S′(S_3); (3) ∆{C_3 } ~(3, 4)
Here, x ∆y denotes that the security device need to be placed between the nodes/node groups x and y.
Following is the output of the above example from the graphviz application. It shows all the connectivity and security device placement of the required pattern. IPSec tunnel is required to be placed between the nodes to carry out the encryption and decryption process.

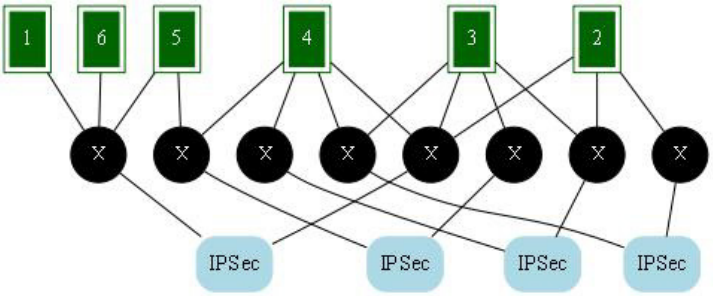

Fig. 2. showing the detailed connectivity and partition pattern of example 2.

4. Implementation and Evaluation

4.1. Setup

Here for our algorithm the platform can be either Windows or Linux, but we have done it in Windows environment. The maximum number of nodes that the algorithm can evaluate efficiently is 100. The algorithm is implemented in 'C' language and the final visualized output is given by the Graphviz application[1]. Considering the combinations of 4 partition patterns as discussed above we are generating an optimised placement of security devices and their respective graphs drawn by the Graphviz application. The details of the IDE (Integrated Development Environment) used for C, i.e. Code::Blocks and the visualization done by Graphviz is explained in below sections.

4.2. Software used

Before we move on with the procedure of placement of security devices, let us look at the various tools used for the designing output of the required network. We need a C compiler, gcc 4.3.8 version. Code::Blocks is a *free C, C++ and Fortran IDE* built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable. Here we have used this tool for writing, editing and compiling our required C program. The result produced is passed to Graphviz application for the visualization of the network along with their security devices placed between them. GVEdit Graph File Editor for Graphvizversion: 1.02 Graphviz: version 2.38.0.

4.3. Time Complexity

We evaluate the device placement with respect to Test cases and time complexity. The algorithms are evaluated in different test networks with varying data center network size and requirements. In addition, evaluation is done considering the separation pattern and offered services. We evaluate the device placement time based on separation

result. It has three major components: (i) separation zone creation based on similar service, (ii) Separation of class creation, and (iii) time for device placement. The time complexity of the both algorithms are $O(N^2)$, where N is the number of computing as well security nodes present in the cloud data center network. This complexity is due to the comparison of every pair of nodes in the data center. Figure 4 shows the relation between device placement time with network size.

4.4. Space Complexity

The space requirements represent the memory used for solving the device placement problem. Figure 5 shows that the memory used linearly varies with the network size under different requirements.

5. Related Work

Several researchers has worked on synthesis and configuration of networks and enterprise networks [2, 3, 4]. There are few works on risk based security configuration analysis. For example, risk analysis using attack graphs have been proposed (in [5, 6]). Others have proposed using attack graphs to find optimal deployment of security devices to block all attack scenarios[7, 8]. However there is no such significant work for cloud infrastructure. In this paper the major thrust is to place security devices appropriately in cloud infrastructure.

6. Conclusion

Although security architecture design usually follows well-known principles such as, partition, defense-in- depth, fail safety etc., but, it is still performed in an ad-hoc manner. Recently, many issues has been raised about the validity and optimality of the data centre security architecture when the design requires balancing of different competing factors such as, partition , cost and usability. Therefore, generating a usable and optimal security configuration resolving the contention between the security requirements and business constraints is an important but challenging problem. In this paper we have presented the security configuration in cloud data centre.

References

1. http://www.graphviz.org/
2. E. Al-Shaer, W. Marrero, A. El-Ataway and K. ElBadani. Network Security Configuration in A Box: End-to-End Security Configuration Verification. In ICNP'09, pp. 123-132, Princeton, NJ, October 2009.
3. L. Yuan, J. Mai, Z. Su, H. Chen, C. Chuah, and P. Mohapatra. FIREMAN: A Framework kit for Firewall Modeling and Analysis. In IEEE Symp. on Security and Privacy ,Oakland, USA, May 2006.
4. S. Majhi, S. Kumar, P.Bera, E. Al. Shaer, M. Satapathy. Synthesizing Optimal Security Configuration for Enterprise Network. In IET System Safety and Cyber Security, Manchester, UK, Oct 2014.
5. X. Ou, W. F. Boyer and M. A. McQueen. A scalable approach to attack graph generation. In 13th ACM CCS, 2006.
6. R. Dewri, N. Poolsappsi, I. Ray and D. Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In 14th ACM CCS 2007.
7. J. Homer and X. Ou. Sat-solving approaches to context-aware enterprise network security management. In IEEE JSAC Special Issue on Network Infrastructure Configuration, 2011.
8. Kotenko and M. Stepashkin. Attack graph based evaluation of network security. In International Conference on Communications and Multimedia Security 2006.