Optimal Placement of Sequentially Ordered Virtual Security Appliances in the Cloud

Alireza Shameli-Sendi[1], Yosr Jarraya[2], Mohamed Fekih-Ahmed[1],
Makan Pourzandi[2], Chamseddine Talhi[1], Mohamed Cheriet[1]

[1]Ecole de Technologie Superieure (ETS), Montreal, Canada
[2]Ericsson Research Security, Montreal, Canada
Email: alireza.shameli@synchromedia.ca, yosr.jarraya@ericsson.com, mohamed.fekih.ahmed@synchromedia.ca,
makan.pourzandi@ericsson.com, chamseddine.talhi@etsmtl.ca, mohamed.cheriet@etsmtl.ca

***Abstract*—Traditional enterprise network security is based on the deployment of security appliances placed on some specific locations filtering, monitoring the traffic going through them. In this perspective, security appliances are chained in specific order to perform different security functions on the traffic. In the cloud, the same approach is often adopted using virtual security appliances to protect traffic for different virtual applications with the challenge of dealing with the flexible and elastic nature of the cloud. In this paper, we investigate the problem of placing virtual security appliances within the data center in order to minimize network latency and computing costs for security functions while maintaining the required sequential order of traversing virtual security appliances. We propose a new algorithm computing the best place to deploy these virtual security appliances in the data center. We further integrated our placement algorithm in an open source cloud framework, i.e. Openstack, in our test laboratory. The preliminary results show that we are placing the virtual security appliances in the required sequential order while improving the efficiency compared to the current default placement algorithm in Openstack.**

I. Introduction

Traditionally, the security appliances (e.g. firewalls, IDS) are designed as monolithic boxes placed on fixed locations in the network and it is of the administrators duties to overload path selection mechanisms to force traffic through the desired sequences of middleboxes as defined within security policies [1]. However, this deployment is rather static and does not fit into the flexible and elastic nature of the cloud. In cloud data centers consisting of thousands of servers, and middleboxes, a huge amount of traffic related to heterogeneous types of applications must be steered through the right sequence of middleboxes. Typically, different types of application needs different types of security. To this end, we propose to consider multiple specialized virtual security appliances (VSA) that can be automatically created and transparently chained and inserted into the paths of the traffic to be inspected to comply with the tenants' needs in terms of security. Additionally, these virtual security appliances need to be efficiently deployed within the data center infrastructure to minimize and balance the overall processing time and the processing power consumption needed for performing different security functions. Therefore, the deployment of such security appliances cannot be performed on ad hoc basis.

After investigating the state of the art, we identified the traveling purchaser problem (TPP) [2] as being a good candidate that closely models our problem and fits into our objectives. There are numerous variants of the TPP [3], [4], however, none of them fully maps into our needs. Thus, we propose in this paper to extend the TPP in two ways. First, we consider multiple independent purchasers instead of a single one. Note that the notion of multiple purchasers existing in the literature differs from our definition. Indeed, existent multi-purchaser variants of TPP model a set of purchasers that cooperate together to buy the same list of unordered products. In our case, we consider independent multiple purchasers where each has a source node to start from, a destination node to end in, and an ordered list of products to be purchased. Second, we add a new feature to TPP, never studied before to the best of our knowledge, which is a sequential order of purchasing the products. We name our extension of TPP, independent multiple purchasers capacitated TPP with order or imCTPP-o. The problem, to the best of our knowledge, has never been investigated before. TPP is known to be NP-hard [3], therefore, heuristics need to be developed for large problems. The rest of this paper is organized as follows: Section II consists of the related work where relevant contributions from the literature are reviewed and compared to our work. Section III provides a mathematical formulation of our optimization problem, namely imCTPP-o. Section IV presents our findings on the adequacy of such an optimization framework and discusses the various experimental results. Finally, in Section V, we present our conclusions and future directions.

II. Related Work

Many works studied the problem of steering traffic through middleboxes placed in the cloud [5]–[9]. But none of those provided a formal approach for optimizing the placement and the ordered sequences of middleboxes. Security placement have been investigated by several works [1], [10]. However, none of the reviewed approaches considered the case of an ordered chain of security appliances. Bouet et al. [10] propose the dynamic placement of Deep Packet Inspection (DPI) engines based on genetic algorithms, while optimizing the cost of DPI engine deployment, minimizing their number, the global network load and the number of unanalyzed flows. The uncapacitated facility location problem is considered to model their optimization problem. Joseph et al. [1] propose a new layer-2 for data centers which has policy-aware switches to control the paths of flows to traverse network security middleboxes

in a particular sequence. In their approach, middleboxes are taken off the physical network path whereas in our proposed approach, each security middlebox is attached to a network appliance.

III. Mathematical Formulation

In the following section, we formalize our variant of TPP, namely imCTPP-o, using integer linear programming. Note that a Security Defense Zone (SDZ) is a logical component that can be attached to a node in the data center where different VSA can be deployed. SDZs can be part of/attached to computing nodes, TOR, aggregation, and core switches.

These are the general variables defined in our problem:

- $M = \{1, \cdots, m\}$: A set of potential sites (markets) that represent the nodes of the network, where some serve for locating the SDZs and others are the source and destination locations for the various flows.
- $P = \{1, \cdots, h\}$: A set of flows (traveling purchasers) that have to traverse the network each from a source node following a path to a given destination node while purchasing a list of ordered VSAs.
- $N = \{1, \cdots, n\}$: A set of virtual security appliances (products) to be purchased at different nodes (except those that are source or destination nodes of the flows).
- S_p (resp. D_p): The source (resp. destination) node of the flow p such that $S_p,\ D_p\ \in\ M$.
- $c_{i,j}$: A positive integer representing the cost of traveling between two nodes i and j, which corresponds to the cost of processing the traffic through that link.
- $b_{i,l}$: A positive integer representing the cost of purchasing a VSA (product) l at market i, which corresponds to the cost of processing the traffic through by VSA l at a node i.
- $d_{l,p}$: The demand of a product l required by the purchaser p such that $d_{l,p}$ is a positive integer.
- $q_{i,l}$: The quantity of product l initially available at market i such that $q_{i,l}$ is a positive integer.
- K_p: The set of VSAs that must be purchased for the flow p.
- $O_{l,p}$: A positive integer that represents the order of purchasing a product $l\ \in K_p$ for the flow $p\ \in\ P$. This is meant to model the order of purchase for each product $l\ \in\ K_p$ for a purchaser p, where $O_{l,p} > O_{r,p}$ means that product r should be purchased before l.

The following are the decision variables:

- $x_{i,j,p}$ denotes whether a market j has been immediately visited after market i by the purchaser p. Thus, $x_{i,j,p} = 1$ if the link (i, j) belongs to the solution for purchaser p and $x_{i,j,p} = 0$ otherwise.
- $y_{i,l,p}$ denotes whether a product l has been purchased at market i by purchaser p. Thus, $y_{i,l,p} = 1$ if the product l is purchased at market i for purchaser p and $y_{i,l,p} = o$ otherwise.
- $v_{i,j,p}$ denotes the order of visiting the link between market i and market j by purchaser p starting from the source node S_p such that $v_{S_p,j,p} = m-1$ if node j is visited immediately after node S_p and $v_{i,j,p}\ >\ v_{k,r,p}$ if the link (i, j) is visited before (k, r).

Our main objective (c.f. Eq 1) is to find an optimal placement of VSAs while considering not only the compute loads on the SDZ but also the load on links through the computed paths. More specifically, we aim at minimizing the processing time needed by security functions, which is composed of the time required by the traffic to traverse the links and the time required by the traffic to be proceeded by the VSAs in the SDZ nodes. Thus, the more the nodes/links are busy (i.e. higher load), the larger would be the cost (i.e. higher processing time). The mathematical formulation of our problem can be stated as follows:

Objective:

$$\textbf{\textit{Min}} \sum_{p\in P}\sum_{i\in M}\sum_{j\in M}(c_{i,j}\cdot x_{i,j,p}) + \sum_{p\in P}\sum_{i\in M}\sum_{l\in N}(b_{i,l}\cdot y_{i,l,p}) \quad (1)$$

Subject to the following constraints:

$$\sum_{j\in M} x_{j,i,p} + (\text{if } i = S_p \text{ then } 1) = \sum_{r\in SDZ} x_{i,r,p} + (\text{if } i = D_p \text{ then } 1); \quad (2)$$

$$\sum_{i\in M} y_{i,l,p} = d_{l,p} \qquad \forall l \in N, \forall p \in P \quad (3)$$

$$\sum_{p\in P} y_{i,l,p} - q_{i,l} \leq 0 \qquad \forall i \in M, \forall l \in N \quad (4)$$

$$\sum_{j\in M} x_{i,j,p} - y_{i,l,p} \geq 0 \quad \forall i \in M \setminus \{S_p\}, l \in N, p \in P \quad (5)$$

$$\sum_{j\in M} x_{i,j,p} \leq 1 \qquad \forall i \in M, \forall p \in P \quad (6)$$

$$v_{i,j,p} \leq m \times x_{i,j,p} \quad \forall i \in M, \forall j \in M, \forall p \in P \quad (7)$$

$$\sum_{j\in M} v_{j,i,p} + (\text{if } i = S_p \text{ then } m) = \sum_{j\in M}(v_{i,j,p} + x_{i,j,p}) \quad \forall i \in M \setminus \{D_p\}, \forall p \in P \quad (8)$$

$$m \times (\sum_{a\in M} x_{a,i,p} - y_{i,k,p}) - \sum_{r\in M} v_{r,i,p} + m \geq m \times y_{j,l,p} - \sum_{r\in M} v_{r,j,p}$$
$$\forall p \in P, i, j \in M \setminus \{D_p\}, i \neq j, l, k \in K_p, l \neq k, O_{l,p} = O_{k,p} - 1 \quad (9)$$

Constraint (2) also known as the flow conservation, denotes that for any node i, the sum of in-degree edges is equal to the sum of out-degree edges, which should be 1 as we are considering a single path per purchaser. For the special case of the source and destination of a given flow, we assume one incoming edge into the source node and one outgoing edge from the destination node. Constraint (3) denotes that the exact amount of products given in the demand list should

be purchased. Constraint (4) states that a product can only be purchased at a given market if it is actually available at that market. Constraint (5) indicates that the node should be included in the solution to be able to purchase a product at that market. Constraint (6) avoids forming cycles in the solution path. Constraint (7) states that each value of $v_{i,j,p}$ should be less then m if $x_{i,j,p} = 1$ or equal zero otherwise. Constraint (8) is used to calculate the values of $v_{i,j,p}$ along the path formed by the solution. Constraint (9) instructs that all products should be purchased in the order it has been requested. More precisely, all products l listed in K_p have to be purchased by p in the order precised in $O_{l,p}$. Thus, if $O_{l,p} = O_{k,p} - 1$, product l should be purchased before product k such that the market i where $y_{l,i,p} = 1$ is visited before the market j, where $y_{k,j,p} = 1$, which is enforced by comparing the values $\sum_{r \in M} v_{r,i,p}$ and $\sum_{r \in M} v_{r,j,p}$ such that $\sum_{r \in M} v_{r,i,p} \geq \sum_{r \in M} v_{r,j,p}$.

IV. Experiments and Discussion

In the following, we present our experimental results and show the feasibility of our approach. We also discuss a preliminary implementation of our algorithm and its integration within an open source cloud framework, i.e. OpenStack [1].

A. imCTPP-o Integration in Cloud Environment

In order show the feasibility of our approach and test it in real Cloud environment, we integrated imCTPP-o in Openstack. To do so, a Cloud Defense Orchestrator (CDO) is designed to interact with openstack control services. CDO collects networking and computing information in Openstack, builds the SDZs Graph with purchasing and distance costs and communicates them to the imCTPP-o engine as input. To realize the service chaining and traffic steering in Openstack, we use OpenDaylight (ODL) [2]. ODL is a SDN controller enabling rich and flexible networking functionalities for CDO. The OpenStack Icehouse version is used with networking plugin mechanism to connect to ODL controller. For imCTPP-o implementation, a solver scheduler in OpenStack Nova scheduler driver is used. We extended Nova scheduler to support our constrained model for VSA placement by invoking our GLPK program implementing imCTPP-o to find the optimal solution. Our CDO is designed to enable a dynamic and optimized VSA placement and orchestration. Our preliminary results show that we are placing VSAs in right sequential order while improving the efficiency of the implementation compared to the current default placement algorithm in Openstack. For lack of space, we consider this work to be subject to a future publication.

B. Simulation Setup

For performance testing and finding the optimal solution, the GLPK (GNU Linear Programming Kit), is used on a machine with 6 cores Intel Westmere (XEON L5638) clocked at 2.30 GHz with 24 GB RAM. We considered a network model with 20 nodes. The traffic and workload patterns may vary dramatically due to divers factors during the life cycle of virtual applications. Therefore, the traveling and purchasing costs are randomly generated in range [0,1]. Furthermore, the quantity of products (VSAs) available at each market (SDZ) was randomly generated in the interval [0,5] with uniform distribution. Higher cost values represent higher workloads for SDZs.

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

[2] http://www.opendaylight.org/

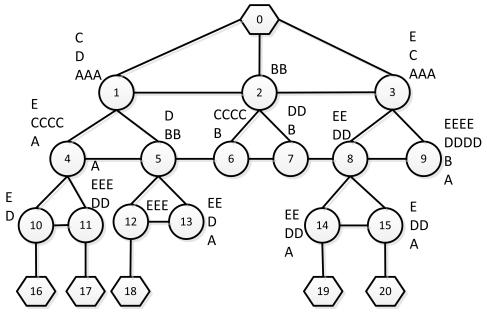


Fig. 1: An example for the SDZ Interconnection Graph and the VSAs Availability in each SDZ.

C. Result

In order to show the feasibility of our approach, a case study is presented. Figure 1 illustrates an example of a SDZ interconnection graph that we are going to use in our experiments later. The graph also shows VSAs availability in each SDZ. For the sake of simplifying the notation, we use the alphabetic letters to denote different types of VSAs. For instance 'A' denotes a VPN endpoint. Let VSAs ={A: VPN, B: FW-L3, C: IDS, D: FWL4-7, E: DPI}. Node n_0 is Internet and nodes $n_{16} \cdots n_{20}$ are four different vApps. Nodes $n_{10} \cdots n_{15}$ are six SDZs within the TOR and nodes $n_1 \cdots n_9$ are nine SDZs within, aggregation switches. We consider four VMs at n_{16}, n_{17}, n_{18}, and n_{20}, respectively. We believe our algorithm can be easily extended by replacing VMs by set of VMs having the same security profile, or vApps or even servers. Our aim is to secure the traffic generated between the Internet node n_0 and these four nodes. Each of the four traffic flows requires a different sequence of virtual security appliances to be placed in a subset of the SDZs within an optimal path. To apply our approach on this scenario, we need to model each traffic flow as an independent purchaser such that each purchaser p_i travels from a source node S_i to a destination node D_i. We have four purchasers that have each these pairs of source and destination nodes, respectively, as follows: $(S_1 = n_0, D_1 = n_{16})$, $(S_2 = n_0, D_2 = n_{17})$, $(S_3 = n_0, D_3 = n_{18})$, and $(S_4 = n_0, D_4 = n_{20})$. The ordered list of products to be purchased by each one is: $K_1 = \{A, B, C, E\}, K_2 = \{A, C, D\}, K_3 = \{A, C, D, E\}, K_4 = \{B, C, E\}$.

Figure 2 illustrates the solution provided by running imCTPP-o using GLPK on the use case presented above. It shows the optimal path calculated for each purchaser, the visited nodes, and the purchased VSAs per node. One can easily note that the sequential order of products purchasing has been respected for the four purchasers in imCTPP-o. Figures 3a and 3b show execution time of our simulations for a network model with 20 nodes according to the number of purchasers, number of products (security appliances), and product availability, respectively. Product availability denotes the total number of instances per VSA type distributed in the model. As seen, when the number of security appliances increases, the algorithm execution time rises exponentially. Indeed TPP

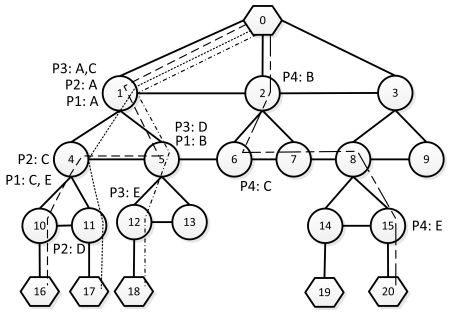
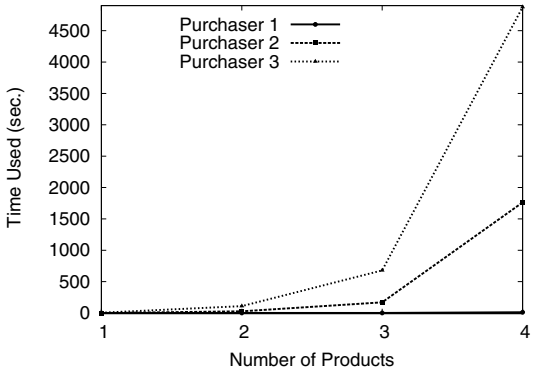
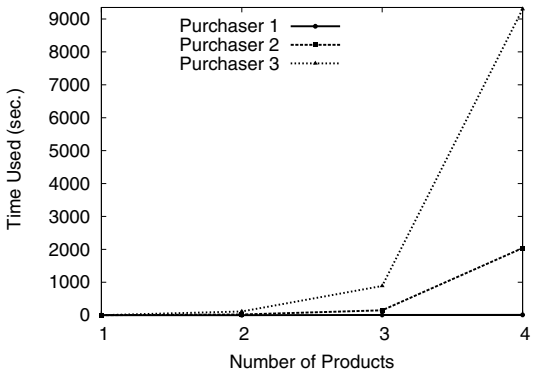


Fig. 2: Result of imCTPP-o



(a) Product availability is equal to 10



(b) Product availability is equal to 20

Fig. 3: Result of imCTPP-o algorithm in a network model with size 20 and different available products for each type distributed in the model.

is NP-hard and we are tackling more complex problem. This complexity is due to the added ordered constraint. From a network of 20 nodes, up to 3 products the optimal solution is calculated in a reasonable time. For product numbers over 4 and number of purchasers over 2, we realize that the execution time goes over 30 minuets. In case of large problems, we will improve our approach using divide and conquer paradigm.

V. Conclusion

With increasing use of cloud for different applications, the security of those applications has received much interest. To meet the security needs of those applications, a large variety of virtual network security appliances are deployed in the cloud. For each application, traffic should go through network security appliances in the pre-ordered sequence to be monitored and filtered. In this paper, we propose a new model, imCTPP-o, aiming to address the placement optimization for ordered sequences of virtual security appliances in the cloud infrastructure. To this end, we defined a new variant of the Traveling Purchaser Problem by considering multiple purchasers with a constraint on the product order by each purchaser. We showed the feasibility and usability of our approach by implementing and integrating the imCTPP-o algorithm in a cloud framework, i.e. OpenStack. We further used GLPK solver to study the correctness of our algorithm for different scenarios. At the end, studied imCTPP-o execution time for different sizes of the problem. As expected from similar research results on TPP problem, for large problem sizes due to the excessive computational times and solver limitations, it is difficult to find an optimal solution in reasonable time. For future research, we consider develop new approach based on divide and conquer paradigm to alleviate this for large problems.

Acknowledgment

This work is partly funded by NSERC Chair on Sustainable Smart Eco-Cloud, NSERC and by the NSERC/ERICSSON CRDPJ: ECOLOTIC Sustainable and Green Telco-Cloud.

References

[1] D. A. Joseph, A. Tavakoli, and I. Stoica, "A policy-aware switching layer for data centers," in *Proceedings of the ACM SIGCOMM 2008 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Seattle, WA, USA*. ACM, 2008, pp. 51–62.

[2] T. Ramesh, "Traveling purchaser problem," *Opsearch*, vol. 18, pp. 78–91, 1981.

[3] L. Gouveia, A. Paias, and S. Voβ, "Models for a traveling purchaser problem with additional side-constraints," *Computers & Operations Research*, vol. 38, no. 2, pp. 550 – 558, 2011.

[4] M.-J. Choi and S.-H. Lee, "The multiple traveling purchaser problem," in *Computers and Industrial Engineering (CIE), 2010 40th International Conference on*, July 2010, pp. 1–5.

[5] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simplefying middlebox policy enforcement using sdn," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*. ACM, 2013, pp. 27–38.

[6] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes." in *NSDI*, 2013, pp. 227–240.

[7] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture." in *NSDI*, 2012, pp. 323–336.

[8] S. K. Fayazbakhsh, V. Sekar, M. Yu, and J. C. Mogul, "Flowtags: enforcing network-wide policies in the presence of dynamic middlebox actions," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 19–24.

[9] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 413–424.

[10] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *Military Communications Conference, MILCOM 2013 - 2013 IEEE*, Nov 2013, pp. 992–997.