Managing the flow of information in the cloud environment

Gesoon J.K. Al-Abbas

Al-Muthanna University, College of Engineering / Iraq
ghusoonjawad@mu.edu.iq

ABSTRACT

Management of information flows in a cloud environment serve as an important information security mechanism. The main problem in the implementation of such a mechanism is the speed of packet switching in accordance with the given rules. The article discusses methods for monitoring and managing information flows in a cloud environment based on software defined networks(SDN). Estimates of the speed of information exchange in the framework of cloud computing are constructed. A system of procedures is discussed that makes it possible to reconfigure switching tables (ST) in accordance with current changes in the topology of network interactions in a cloud environment.

Keywords: cloud computing, management of information flows, network change management, switching tables.

1. Introduction

The need to virtualize three basic resources—computing, storage, and networking—responded to a range of new approaches, such as Software Defined Networks (SDN) [1] and Network Function Virtualization (NFV) [2], embodied in effective applied information technology solutions of a new generation. They were included in the Internet projects [3] and the American national program GENI (Global Environment for Network Innovations) [4], which is currently developing as an international cooperation.

Critical characteristics of cloud solutions turned out to be the speed and security of information exchange. At the same time, these two indicators turn out to be "oppositely directed". A high level of security, as a rule, significantly reduces the speed of cloud interactions, as it reduces the speed of data exchange. On the contrary, high speeds of information exchange impose very stringent requirements on the architecture and functionality of the corresponding information security systems.

Traditionally, security solutions in large information systems are based on access control. For example, an RBAC (Role Based Access Control) security policy [5] can be built using isolated domains that correspond to the privilege tree in the organization.

The second most important mechanism for ensuring information security can be the control and management of information flows .

Note that with the help of SDN, information flows can be controlled and managed. With information flow control, you can create isolated domains. Thus, SDN can provide critical tamper protection functions.

Another important function in the security system of an information system is the monitoring of security events [6]. One of the most effective mechanisms for solving this problem is DPI (Deep Packet Inspection). Note that DPI at the SDN controller level also solves the problem of protecting the monitoring system.

Here are some technological solutions that allow SDN to solve the above tasks of ensuring the information security of cloud computing. Due to the special division of the level of execution of data traffic and the level of control of this traffic into two areas, it is possible to form additional possibilities for managing network interactions compared to using classical network technologies. Thus, in particular, it is possible to obtain significant advantages by transferring a number of important functions to the SDN controller, which (as a software system that provides virtualization of the necessary functions for managing a computer network) can be located on a separate (including distributed) computer installation with the required performance. As a result, opportunities for flexible control of the calculations performed on the controller and security are opened up.

Interest, first of all, are the possibilities to transfer a number of important, but secondary in relation to the actual transmission of packets and data streams, computational procedures to an external controller in relation to the traffic organization mechanisms. This allows you to implement a number of resource-intensive traffic control procedures outside the traffic organization mechanism (outside the SDN switch), using an external computer with the required performance. Let's call it the SDN controller. This technique allows us to divide the solution of a number of tasks of organizing and managing cloud traffic, which are "sensitive" to the volume of required calculations, into two parts:

1. one-time execution of a special part of an integral set of procedures for analyzing and managing traffic, which requires significant amounts of calculations.
2. a repetitively performed set of actions that are quite low-cost in terms of the volume of calculations that implement them.

The first of them can be executed on an external SDN controller, and the second one can also be performed on an SDN switch.

As an illustrative example of such separation, we can refer to the problem of searching for the header of an input packet in the ST (this task will be discussed in detail in the following sections). A fairly simple formal procedure is to be executed here - to check whether the input Boolean vector of fixed length k is contained in the given ST containing n rows of Boolean vectors of the same length k. However, when checking such occurrences, the following essential considerations must be taken into account. In real applications, ST turn out to be very large. At the moment, industrial switches offered to the market are known, operating with ST with a total size of 150-200 thousand lines. In this case, a high speed of solving the problem is required.

When solving this formal problem, one can use a number of well-studied options for dividing it into two sub problems, one of which will be solved once, generating a reduction of the ST to some special form and requiring significantly larger (compared with the solution of the second sub problem) amount of calculations . The second subtask will provide multiple quick checks, using this special kind of ST, of the actual embedding of the input header h in the current state of the ST[7].

Well-known examples of such divisions are:

– lexicographic ordering of the initial ST, when checking the embed ability of h in the ST (as well as introducing dynamic changes to the current state of the ST - which will be discussed a little later) turns out to be a fairly simple computational procedure.

– reduction of the ST to the form of a binary tree (diagram), when the generation of such a tree turns out to be, in the general case, an exponentially complex computational procedure, and the actual check of the embedding of the input header h in an already constructed tree is a procedure that requires only a linearly rapidly growing vector h with increasing length amount of calculations.

However, when moving from the analysis of the headers of the transmitted data packets to the analysis of their content (DPI, IDS / IPS (Intrusion Detection / Protection Systems)), the situation under consideration becomes much more complicated. The set of features checked for occurrence in the "body" of the transmitted packets increases many times. For example , actual virus signature databases today already operate with hundreds of thousands of objects. At the "technical" level, this problem of "dimension" of the object of analysis is reflected in the fact that :

- its formalization in the “technique” of operations with boolean vectors already presented above leads to the need to work with objects of very large length, containing significantly more zeros than ones.
- in such Boolean vectors, all useful information is already represented by elements of unordered sets.

Some variants of a static mechanism for controlling network traffic using the capabilities of the SDN approach were already considered[8]. In this paper, we propose a purposeful extension of the presented approach to the case of managing dynamic changes in the actual configuration of the SDN network, which takes into account, in addition to processing the headers of data packets, the need for organizing a quick analysis of their content. At the same time, as before, the main role in the proposed procedural design is assigned to the effective organization (on the SDN controller functioning according to the proposed special rules) of auxiliary calculations, which provide support for decision-making on reconfiguration in the corresponding information exchange network.

2. Methods for checking the embedding of vectors in the switching table

In industrial applications, the "width" of a ST (the number of its columns) k, as a rule, is significantly less than the number of its rows n. However, keep in mind that $k \leq \log_2 n$. Note that a direct search, i.e., iterating over the lines of the switching table, can also be optimized by using lexicographic orderings of the lines of the original ST or, for example, by reducing the original ST to the form of a binary tree. Let's consider these considerations in more detail.

Let L(ST) be a lexicographically ordered by row version of the analyzed ST. Let us denote the number of operations for the lexicographic ordering of a ST by S^1_L. It is easy to see that the procedure for reducing a ST to the target form requires the generation of L(ST) $S^1_L \leq C^1_{\mathrm{L}}$ $kn\log_2 n$ operations, where $C^1_L = const$. Thus, the following theorem holds.

<u>Theorem 1.</u> The complexity of reducing to the form L(ST) a switching table with n rows and k columns is bounded above by $C^1_{\mathrm{L}}\ kn\log_2 n$.

When checking the nesting of the input header $hi = \{a^1_{i1}, a^2_{i2}, \ldots, a^k_{ik}\}$ in L(ST) in each of its columns with index $j + 1$, $j \in \{1, 2, \ldots, k - 1\}$, one will have to search for the corresponding pair of “blocks” (sets of rows ST) containing only 1 and only 0, into which the set of values a^j_{ij} “splits” in the previous column number j. To reduce the amount of enumeration of values in each of the columns K (compared to direct viewing of each column as a whole), you can use the algorithm of sequential partitioning of the sets of column elements, for example, in half or according to the golden section rule. However, However, the number of such blocks with increasing number j will grow as $\log_2 j$, i.e.,no faster than $C^2_L\log_2 n$ for some constant C^2_L.

Let S^2_L be the size of the enumeration in checking the embeddability of the heading h in L(ST). Thus, the following theorem holds.

<u>Theorem 2.</u> The amount of enumeration in checking the embeddability of the header h in L(ST) of the switching table can be estimated as $S^2_L \leq C^2_L\log_2 n$.

Let us define a binary tree corresponding to ST. The root of the tree is at level (layer) 0, and no more than two edges labeled 0 and 1 can go from each vertex to the next layer. Each vertex, except the root, has label 0 or 1, corresponding to the label of the incoming edge. Thus, each path from the root to the vertex of layer i corresponds one-to-one to a binary vector of length i (both in terms of edge labels and vertex labels). The number of layers in the binary tree representing the ST is k. The maximum number of vertices in layer k is 2^k, and the number of all vertices in the complete binary tree is $2^{k+1} - 1$.

We select paths in a complete binary tree with k layers corresponding to vectors from ST. The resulting tree will be denoted by D(ST). Checking the presence of a vector in a ST using the constructed tree D(ST) consists in passing a path of length k, and from each vertex the transition is carried out along an edge with a label corresponding to the next sign in the checked vector. If at the next step the required edge is absent, then this means that the considered vector does not exist in the ST. Note that, by the construction of the tree D(ST), if the ST is not empty, there is at least one outgoing edge from any reached vertex. Checking the presence of vectors in a ST using the tree D(ST) does not exceed k steps.

To construct D(ST) on some layout of a complete binary tree, $2^{k+1} - 1$ memory elements and nk fragment allocation operations D(ST) in this layout are required.

Theorem 3. Determining whether a vector belongs to the switching table using D(ST) requires no more than k operations of traversing the edges of D(ST), but the construction of D(ST) requires nk operations of extracting fragments of D(ST) in a complete binary tree layout with k layers.

Let $k = \alpha\log_2 n$, where $\alpha \geq 1$. Then the memory size for the full binary tree layout is $2n^{\alpha} - 1$. The complexity of constructing D(ST) is equal to $\alpha n\log_2 n$, and checking membership in a ST will require no more than $\alpha\log_2 n$ operations.

The binary tree method can be improved in such a way that the speed of checking embedding will be significantly reduced with minor deterioration of the remaining parameters.

In a binary rooted tree D(ST), we mean the least forbidden path (respectively, a vector) from the root that ends with a missing edge. A ban is a path (vector) corresponding in D(ST) to the missing path to the terminal node at level k. For every prohibition in a complete binary tree, there is a smallest prohibition. The embeddability of a vector in a ST can be checked by the tree of least prohibitions M(ST). If the beginning of the checked vector coincides with the corresponding least prohibition in the tree M(ST), then the checked vector does not lie in the ST.

On the simplest model, let us estimate the influence of short and long least prohibitions. Assume that x is a vector to be checked, and the ST is generated randomly using an independent equiprobable choice of vectors of length k. Denote by P_i the probability that a random vector will collide with x up to the coordinate with index i. Then $1 - P_i$ means the probability that the random vector coincides with the vector x in all i first coordinates, while $1 - P_i = 1/2^i$. The probability that the difference of vectors will occur at the coordinate i is equal to $2^{-(i+1)}$. The number of vectors for which the difference occurs in the coordinate with index i is distributed according to the binomial law. This implies that the average number of vectors out of n independent random vectors in which a difference occurs at step $i + 1$ is equal to $n/2^{i+1}$. This number decreases as i increases, which serves as an argument in favor of the fact that the main number of the smallest prohibitions has a small length. These considerations show the expediency of using the tree of least prohibitions to determine the membership of the ST vector.

The speed of determining the membership of the ST vector using binary trees can be increased by parallelizing the search for the least prohibitions. To this end, consider a pair of complete rooted binary trees with k layers, which we will call "left" and "right". The switching table generates a subtree D_L(ST) in the left complete tree, as was described above when constructing D(ST). Similarly, on the basis of the right complete tree, the tree D_R(ST) is constructed from the ST. The difference between the trees is that the paths in D_R(ST) correspond to vectors from ST, read from right to left. Accordingly, two least prohibition trees M_L(ST) and M_R(ST) are constructed. If the vector x is not in ST, then there are two paths in M_L(ST) and M_R(ST) corresponding to the vector x when it is read from left to right in M_L(ST) and from right to left in M_R(ST). The vector x corresponds to the least prohibitions x_L and x_R. Searching for the least prohibition for the tested vector x in parallel using M_L(ST) and M_R(ST) can speed up the rejection process.

If there are other signs of rejection (requirements of the security policy), several trees of the least prohibitions can also be considered in parallel, which makes it possible to speed up the process of resolving the information flow.

3. Switching table reconfiguration procedure

Let at the moment of time T the ST had the configuration $H(T)$. Let us assume that after the time interval ΔT_1, the first structural changes occurred in the table:
– rows (headers) that form the set $H^-(T + \Delta T_1)$ have been removed;
– added rows forming the set $H^+(T + \Delta T_1)$.
Moreover, from a formal point of view, it seems natural to assume that at least one of the sets $H^+(T + \Delta T_1)$ and $H^-(T + \Delta T_1)$ is not empty. In this way

$$H(T +\Delta T_1) = (H(T)\backslash H^-(T +\Delta T_1))\cup H^+(T +\Delta T_1)$$

The exclusion of vectors or the inclusion of new ones in a lexicographically ordered ST in terms of complexity is tantamount to searching for vectors in such a table. This remark determines the complexity of inserting or removing new vectors from a lexicographically ordered ST.
Deleting a path from a tree is equivalent to deleting the nearest root edge and all its extensions. Adding a path to a tree is the same as adding new edges. Hence the complexity of this change does not exceed k.

4. Control and management of information flows by the considered methods

It is easy to see that content monitoring and data flow management in the cloud environment can be based not only on the analysis of the structure of headers (boolean vectors of fixed length). In particular, one can also take into account certain characteristics of the content of the transmitted packets. In ST, the processing of headers was discussed, where it was necessary to operate with vectors of a fixed and at the same time sufficiently small in comparison with the size of the data packet of length. In the case of taking into account the content characteristics of packages, it becomes necessary to process subsets of, generally speaking, a sufficiently large set of features, for example, a set of content characteristics of packages allocated by DPI or IDS / IPS tools. Thus, it becomes possible to process many information security parameters in a single process.
All these features can be assembled into some description template, where the presence of a particular feature in a particular data packet will be encoded by one, and its absence by zero. However, the size of such a template (the number of positions in the Boolean vector corresponding to this template) may turn out to be large.
The presented possibilities can be very useful in optimizing the functioning of the information security subsystem in the cloud landscape, which is responsible, for example, for monitoring the implementation of the relevant computer network security policy, including control over:

1. "non-overlapping" of the specified cloud subnets (i.e., the physical separation of the system and technical resources used by each of the two corresponding subnets, if this is given by the security policy, with dynamic changes in the topology of the cloud virtual infrastructure taking place).
2. availability (including compliance with a given security policy) of specific network interactions of the type (node X - node Y) or (set of nodes Q - set of nodes R).
3. the availability of packages of a given type only in a predetermined list of nodes.

With dynamic changes in the current topology of the cloud network virtual architecture, it is not enough just to monitor (and stop) possible violations of predefined security policy requirements. For example, function (3) can be supported by means of DPI or IDS / IPS solutions, which are convenient to run on the SDN controller. It is equally important to perform a fast, i.e., not generating critical traffic delays in the network, reconfiguration of the ST to a form that provides a solution to current problems in a given cloud landscape and at the same time satisfies the requirements of a pre-formulated security policy.

5. Conclusion

In order to take advantage of cloud storage and the security and speed of data flow, this requires the proposed procedures to manage the ST reconfiguration in all cases and accurately, which will take into account the functions available in the current system of the cloud environment, to determine the

calibration values, it is necessary to conduct many test experiments on computer equipment, There is also another direction and additional features which are the possibilities of balancing the accounts in the implementation of the proposed actions to reconfigure the switching schedules.
Finally, the study of those additional features provided by the proposed approach to managing cloud computing resources to document the "state" and "authorities" of transmitted information packets, as well as control the validity of their addresses as important functions for maintaining the stability and security of information exchange in the cloud computing environment.

References

1. tellectual History of Programmable Networks," SIGCOMM
2. Comput. Commun. Rev., vol. 44, no.2, Apr. 2014, pp. 87–98.
3. Intellectual History of Programmable Networks," SIGCOMM
4. Comput. Commun. Rev., vol. 44, no.2, Apr. 2014, pp. 87–98.
5. N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN: An Intellectual History of Programmable Networks," SIGCOMM Comput. Commun. Rev., vol. 44, no.2, Apr. 2014, pp. 87–98.
6. ETSI. Network functions virtualization. http://www.etsi.org/technologies-clusters/technologies/nfv.
7. The Internet2 community: Enabling the future. http://www.internet2.edu.
8. GENI: Exploring networks of the future. https://www.geni.net.
9. Ferraiolo D. F., Kuhn D. R. Role-based access controls //15th National Computer Security Conference. Baltimore, 1992. P. 554–563. http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf.
10. Becchi M, Franklin M, Crowley P.A workload for evaluating deep packet inspection architectures. In: Proceedings of the 11th IEEE international symposium on workload characterization (IISWC 2008), Seattle, WA, September, 2008.
11. Zhu, Z. Q., Ren, Y., and Liu, J. M.: 'Improved torque regulator to reduce steady-state error of torque response for direct torque control of permanent magnet synchronous machine drives,' IET, Electr. Power Appl., 2014.
12. Fernandes, S.; Kamienski, C.; Szabó, G.; Westholm, T. Deep packet inspection tools and techniques in commodity platforms: Challenges and trends. J. Netw. Comput. Appl. 2012.