Securing the Network Topology in a Source Routing Multi Domain SDN

Sarat Chandra Prasad Gingupalli
NITK,Surathkal

Saumya Hegde
NITK,Surathkal

ABSTRACT

In this paper we look at the research work on hiding the topology details in a source routing multi domain software defined networking(SDN), in which each domain is handled by a single controller and all are coordinated by a central controller. Secrecy of the topology is maintained by the secret key shared between the central controller and the other controllers. The central controller redistributes the shared secret key among other controllers once if the threshold number of flows are reached.

General Terms:

Shortest path, Peripheral nodes, Articulation points

Keywords:

Software Defined Networking(SDN), Source Routing, Encryption

1. INTRODUCTION

The Internet is one of the great technology success stories of the twentieth century. It has enabled greater access to information, provided new modes of communication among people and organizations and has fundamentally changed the way we work, play and learn. It has bought the entire world under a single roof. Unfortunately, the Internet's very success is now creating obstacles to innovation in the networking technology that lies at its core and the services that use it. The size and scope of the public Internet now make the introduction and deployment of new network technologies and advanced services difficult. While the research community has developed innovative solutions to a wide range of networking challenges, there has been remarkably little progress towards deploying these capabilities in the Internet at large.
The ossification of the Internet is a natural evolutionary stage in the development of any highly successful technology. To some extent these problems are addressed by network virtualization, but because of the overhead involved causing hindrance to the growth of technology. To address these issues the concept of programmable networking was introduced. Software defined networking is one such programmable networking technology which aim to provide network as a service like any other resources such as computing and storage. Because of its dynamism in implementing new network protocols and configuration of network it as gained a very popularity in less time.
Revealing the network topology in a multi tenant environment can cause the entire network shutdown if the network topology is misused. In case of cloud multi tenant networking environment one vendor wants to hide the details of their infrastructure from the other vendor.
By identifying the articulation points and cut vertex set in the network, the attacker can implement active attacks like breaking down the communication links in the network and passive attacks like eavesdropping over the communication channel.
In this paper we used a random decomposition technique to decompose the given network graph and on top of it we applied the encryption algorithms to maintain the secrecy of the computed path.

2. LITERATURE SURVEY

As SDN is a new design paradigm so the security challenges of SDN have to be addressed from the scratch. As the control plane is separated from the data plane in SDN, it is creating a clear centralized point of attack. Because of heterogeneity in the traditional networks an attacker have to follow different attacking strategies in order to attack the entire network but where as in the case of SDN, entire network can be compromised by compromising the controller.
Some new threats and attacks on SDN have been proposed in recent time and active research is going to address the security challenges in SDN. An attack on SDN is proposed in [3], which is implemented by fingerprinting the SDN network and then launching efficient resource consumption attacks such as DOS attack on control and data plane. The possible threats to SDN along with counter measures are studied in [4]. A secure controller platform named as permOF is defined in [5] which manages communication of OF applications securely with the SDN controller.
One common thing in all the proposed attacks and threats in [3], [4] and [5] are they are independent of the routing algorithm is being used. In this paper the we presented the possible attacks on multi domain source routing SDN along with their countermeasures. In source routing the attacker have an advantage of knowing the network topology, once if the network topology is known the attacker can implement the further attacks such as bypassing the flow or DOS attacks on data plane.. By using encryption and secret key sharing among the controllers an approach is proposed in this paper to hide the network topology.

3. PROPOSED APPROACH

Let $G = (V, E)$ be a given network graph, where V is the set of switches, E be the set of communication links among them and K be the number of controllers.

3.1 Initial Setup

The steps to be followed for the initial decomposition of the given network graph and assigning switches to the controllers are as follows:

a) By applying any of the random graph decomposition techniques or the approaches proposed in [1] and [2], decompose the given network into $k-1$ components and we call those components as domains.

b) Allocate one controller per domain and assign all the switches in that domain to that controller and we call these controllers as local domain controllers or slave controllers which are coordinated by a master controller.

c) Assign all the switches to master controller to provide the visibility of entire network to the master controller, which takes the responsibility in hiding the topology details of domain to another.

d) Each slave controller applies Warshall Floyd algorithm on all the switches to compute the shortest paths between them.

e) Master controller applies Warshall Floyd algorithm on all the edge switches to compute the shortest paths between them.

3.1.1 Graph Decomposition. Let $G = (V, E)$ is decomposed into $k-1$ components $C_1, C_2, ..., C_{k-1}$.
The i^th component C_i can be represented as $C_i = (V_i, E_i)$. Where V_i is the combination edge and non-edge switches. Edge switches are the ones which establishes the inter domain communication and are linked to both local domain controller and master controller.

$$V_i = V_{ei} \cup V_{nei}$$

Where V_{nei} and V_{ei} are the set of non edge and edge switches of the i^th component C_i respectively. $k_1, k_2, ..., k_{k-1}$ be the shared secret key between the master controller and the corresponding local domain controllers $1, 2, ..., k-1$ respectively.

3.2 Path Generation

Let s be the source switch and d be the destination switch, the master controller computes shortest path between s and d as follows:

(1) Identify the source and destination switch domains.
(2) Obtain edge switches of domains to which source and destination switches belongs
(3) Construct the network graph with source, destination and the corresponding domain edge switches.
(4) Apply the Dijkstra's algorithm on the above graph to find the shortest path between source and destination switches.

Let $C_s = (V_s, E_s)$ and $C_d = (V_d, E_d)$ be the domains such that

$$s \in V_s, d \in V_d$$

and

$$V_s = V_{es} \cup V_{nes}, V_d = V_{ed} \cup V_{ned}$$

$G^{'} = (V^{'}, E^{'})$ be the graph constructed by the master controller, such that

$$V^{'} = s \cup V_{es} \cup V_{ed} \cup d$$

Then Dijkstra's algorithm is applied on the graph $G^{'} = (V^{'}, E^{'})$ to find the shortest path between the source and destination switches s and d respectively.

The computed path is a combination of a source switch, followed by a set of edge switches, which further followed by a destination switch and path will be represented as follows:

$$P = \{s, intermediate\_switch \in \{V_{es} \cup V_{ed}\}, d\}$$

The master controller retrieves the shortest paths between the edge switches which belongs to same domain by communicating with the slave controller.

3.3 Complete Path Generation and Encryption

Two approaches are proposed in this paper for generation of encrypted complete path in this paper.

(1) Encrypting the sub paths of a domain using it's corresponding shared secret keys
(2) Encrypting the complete path using the all shared secret keys

3.3.1 Encrypting the sub paths using corresponding shared secret keys. The algorithm to retrieve the complete path from P and encrypting it is done as follows:

Data: P
Result: Complete Path CP
initialization;
while *not at end of P* **do**
  read current_switch;
  read next_switch;
  retrieve path from the corresponding slave controller;
  encrypt the retrieved path using the secret key shared between the master and the corresponding slave controller of the domain to which current_switch and next_switch belongs to by a master controller;
  and add it to CP;
  go back to the beginning of current section;
end
Algorithm 1: Algorithm to generate and encrypt the complete path

3.3.2 Encrypting using all shared secret keys. In this first complete path CP is generated from P and further which is encrypted by all the shared secret keys from which the CP passes on. The algorithm to generate the encrypted complete path is as follows:

Data: P
Result: Complete Path CP
initialization;
while *not at end of P* **do**
  read current_switch;
  read next_switch;
  retrieve path from the corresponding slave controller;
  add it to CP;
  and encrypt CP excluding the first switch using the secret key shared between the master and the corresponding slave controller of the domain to which current_switch and next_switch belongs to by a master controller;
  go back to the beginning of current section;
end
Algorithm 2: Algorithm to generate the encrypted complete path using all shared secret keys

3.4 Retrieval of Encrypted Path for Communication

Once if the encrypted complete path is generated by the master controller, the communication among the switches between different domains in the above mentioned approached is describes as follows:

3.4.1 In the case where only sub paths are encrypted using the corresponding controller shared secret key. In this approach cryptic flow tables will be maintained in the switch by a slave controller in it's corresponding domain switches. A cryptic flow table is the one which maps the port number to it's corresponding encrypted value.

3.4.2 In the case where complete sub path is encrypted using all shared secret keys through the complete path passes on. In this approach whenever a packet enters into a new domain the source route in the header is encrypted using the corresponding shared secret key of that domain.

3.5 Identifying threshold value and redistributing the shared secret keys

Let $G = (V, E)$ be the given graph and is decomposed into $k-1$ components $C_1, C_2, ..., C_{k-1}$. The number of possible flows in the entire graph is equal to

$$\prod_{v \in V} deg(v)$$

The number of possible flows in a given i^{th} component $C_i = (V_i, E_i)$ is equal to

$$\prod_{v \in V_i} deg(v)$$

By observing different $\prod_{v \in V} deg(v)$ flows the attacker can derive the entire network topology and by observing the $\prod_{v \in V_i} deg(v)$ flows passes through a particular domain an attacker can derive the topology of that domain. In order to prevent the attacker to derive the topology of a network, the central controller assigns a new set of key values to the slave controllers anytime before any of the threshold values reached.

4. WORKING EXAMPLE

Consider the below multi domain SDN network containing of switches $s_1, s_2, ..., s_10$ and four controllers c_1, c_2, c_3 and c_4 is decomposed into three domains. The switches assign to the controller are as follows

$$c_1 = \{s_1, s_2, s_3, s_4\}$$
$$c_2 = \{s_5, s_6, s_7\}$$
$$c_3 = \{s_8, s_9, s_{10}\}$$
$$c_4 = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}\}$$

Where

$$Edge\_switches = \{s_3, s_4, s_5, s_7, s_8\}$$
$$Non\_edge\_switches = \{s_1, s_2, s_6, s_9, s_{10}\}$$

1 In this example $c_1, c_2$and c_3 be the slave controllers, where as c_4 is the master controller. k_1, k_2, k_3 be the shared secret keys of the slave controllers c_1, c_2 and c_3 respectively. Let s_1 be the source switch and s_{10} be the destination switch.

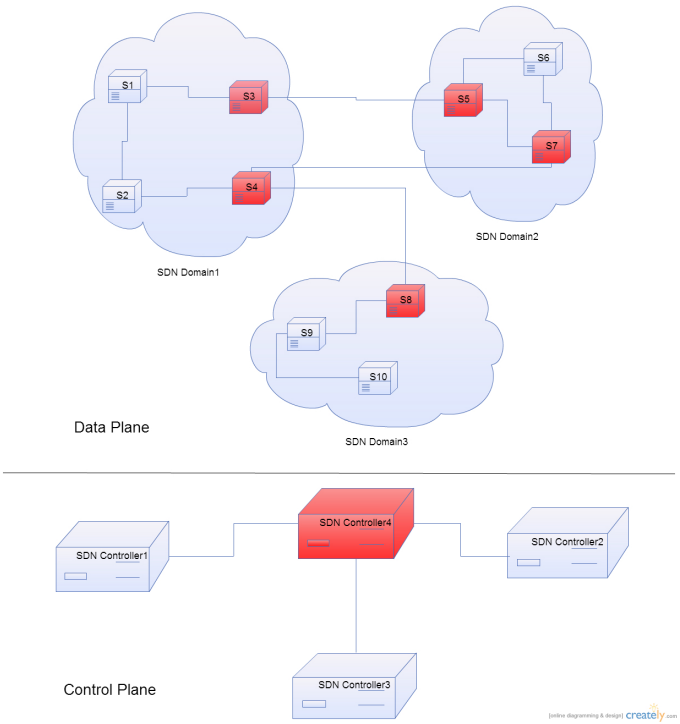

Fig. 1. Multi Domain SDN Network

4.1 Initial Setup

The initial computation of shortest paths is same in both proposed approaches. The controllers $c_1, c_2 and c_3$ will compute shortest paths between all the switches which are assigned to them by applying the well known all pair shortest path algorithm such as Warshall Floyd.
In the same way the master controller c_4 will compute the shortest paths between edge switches $\{s_3, s_4, s_5, s_7, s_8\}$ which are assigned to it.
Once if the computation of shortest paths are done, the source controller c_1 communicates with the master controller c_4 to get the complete complete encrypted shortest path.
First the master controller computes the shortest path interms of edge switches and is done as follows: As master controller has the visibility to entire network topology it identifies that
$s_1 \in c_1$, $s_2 \in c_3$ Edge switches of c_1 are $\{s_3, s_4\}$,
Edge switches of c_2 are $\{s_6, s_7\}$
the edge switches of c_3 are $\{s_8\}$
The master controller constructs a graph $G^{'} = (V^{'}, E^{'})$ such that

$$V^{'} = \{s_1, s_3, s_4, s_6, s_7, s_8, s_{10}\}$$

Dijkstra's algorithm is applied in $G^{'} = (V^{'}, E^{'})$ and the computed shortest path will be

$$P = s_1, s_3, s_6, s_7, s_4, s_8, s_{10}$$

The generation and retrieval of complete encrypted shortest path CP from P in both the proposed approaches is explained as follows:

4.2 In the case where only sub paths are encrypted using the corresponding controller shared secret key

4.2.1 Generation of complete encrypted path

(1) Initially $CP = NIL$ and $P = s_1, s_3, s_6, s_7, s_4, s_8, s_{10}$

(2) Read the switches s_1, s_3, $path(s_1, s_3) = \{s_1, s_3\}$. Encrypt $path(s_1, s_3)$ using k_1 and add it to CP

$$CP = CP \cup \{E(s_1, k_1), E(s_3, k_1)\}$$
$$CP = \{E(s_1, k_1), E(s_3, k_1)\}$$

(3) Read the switches s_6, s_7, $path(s_6, s_7) = \{s_6, s_7\}$. Encrypt $path(s_6, s_7)$ using k_2 and add it to CP

$$CP = CP \cup \{E(s_6, k_2), E(s_7, k_2)\}$$
$$CP = \{E(s_1, k_1), E(s_3, k_1), E(s_6, k_2), E(s_7, k_2)\}$$

(4) Read the switches s_4, and encrypt it using k_1 and add it to CP

$$CP = CP \cup \{E(s_4, k_1)\}$$
$$CP = \{E(s_1, k_1), E(s_3, k_1), E(s_6, k_2), E(s_7, k_2), E(s_4, k_1)\}$$

(5) Read the switches s_8, s_{10}, $path(s_8, s_{10}) = \{s_8, s_9, s_{10}\}$. Encrypt $path(s_8, s_{10})$ using k_3 and add it to CP

$$CP = CP \cup \{E(s_8, k_3), E(s_9, k_3), E(s_{10}, k_3)\}$$
$$CP = \{E(s_1, k_1), E(s_3, k_1), E(s_6, k_2), E(s_7, k_2), E(s_4, k_1), E(s_8, k_3), E(s_9, k_3), E(s_{10}, k_3)\}$$

4.2.2 Communication between the switches. In this approach the slave controller installs the flow tables into it's corresponding switches, the flow tables in each switch map the actual outgoing interface port to it's corresponding encrypted value. The the given example, the flow tables of the switches are shown below

Flow table of switch s_1

Actual value	Encrypted value
s_2	$E(s_2, k_1)$
s_3	$E(s_3, k_1)$

Flow table of switch s_2

Actual value	Encrypted value
s_1	$E(s_1, k_1)$
s_4	$E(s_4, k_1)$

Flow table of switch s_3

Actual value	Encrypted value
s_1	$E(s_1, k_1)$
s_5	$E(s_5, k_2)$

Flow table of switch s_4

Actual value	Encrypted value
s_2	$E(s_2, k_1)$
s_7	$E(s_7, k_2)$
s_8	$E(s_8, k_3)$

Flow table of switch s_5

Actual value	Encrypted value
s_3	$E(s_3, k_1)$
s_6	$E(s_6, k_2)$
s_7	$E(s_7, k_2)$

Flow table of switch s_6

Actual value	Encrypted value
s_5	$E(s_5, k_2)$
s_7	$E(s_7, k_2)$

Flow table of switch s_7

Actual value	Encrypted value
s_4	$E(s_4, k_1)$
s_5	$E(s_5, k_2)$
s_6	$E(s_7, k_2)$

Flow table of switch s_8

Actual value	Encrypted value
s_4	$E(s_4, k_1)$
s_9	$E(s_9, k_3)$

Flow table of switch s_9

Actual value	Encrypted value
s_8	$E(s_8, k_3)$
s_{10}	$E(s_{10}, k_3)$

Flow table of switch s_{10}

Actual value	Encrypted value
s_9	$E(s_9, k_3)$

Whenever a packet comes to the switch it maps the encrypted address to the actual address and forwards the packet.

4.3 In the case where complete sub path is encrypted using all shared secret keys through the complete path passes on

4.3.1 Generation of complete encrypted path

(1) Initially $CP = NIL$ and $P = \{s_1, s_3, s_6, s_7, s_4, s_8, s_{10}\}$

(2) Reverse P, then $P = \{s_{10}, s_8, s_4, s_7, s_6, s_3, s_1\}$

(3) Read the switches s_{10}, s_8, $path(s_{10}, s_8) = \{s_{10}, s_9, s_8\}$. Add $\{s_9, s_{10}\}$ to CP, encrypt CP using k_3 and add s_8 to CP.

$$CP = \{s_9, s_{10}\} \cup CP$$
$$CP = E(\{s_9, s_{10}\}, k_3)$$
$$CP = \{s_8\} \cup CP$$
$$CP = \{s_8, E(\{s_9, s_{10}\}, k_3)\}$$

(4) Read the switch s_4, add $\{\}$it to CP, encrypt CP using k_1 and add s_4 to CP.

$$CP = \{\} \cup CP$$
$$CP = E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1)$$
$$CP = \{s_4\} \cup CP$$
$$CP = \{s_4, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1)\}$$

(5) Read the switches s_7, s_6, $path(s_6, s_7) = \{s_6, s_7\}$. Add $\{s_7\}$ to CP, encrypt CP using k_2 and add $\{s_6\}$ to CP.

$$CP = s_7 \cup CP$$
$$CP = E(\{s_7, \{s_4, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1)\}\}, k_2)$$
$$CP = \{s_6\} \cup CP$$
$$CP = \{s_6, E(\{s_7, \{s_4, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1)\}\}, k_2)\}$$

(6) Read the switches s_3, s_1,$path(s_1, s_3) = \{s_1, s_3\}$. Add $\{s_3\}$ to CP, encrypt CP using k_1 and add $\{s_1\}$ to CP.

$$CP = \{s_3\} \cup CP$$
$$CP = E(\{s_3, \{s_6, E(\{s_7, \{s_4, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1)\}\}, k_2)\}\}, k_1)$$
$$CP = \{s_1\} \cup CP$$
$$CP = \{s_1, E(\{s_3, \{s_6, E(\{s_7, \{s_4, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1)\}\}, k_2)\}\}, k_1)$$

4.3.2 Communication between the switches. In this approach whenever a packet comes to an edge switch from the neighboring domain it decrypts the source route by it's corresponding shared secret key.
In the above example the path received by source switch s_1 is

$$CP = \{s_1, E(\{s_3, s_6\}, E(\{s_7, s_4\}, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1), k_2), k_1)\}$$

Once after receiving the path the source switch sends $CP - \{s_1\}$ to c_1, then c_1 decrypts $CP - \{s_1\}$ using k_1.

$$D(CP - \{s_1\}, k_1) = D(E(\{s_3, s_6\}, E(\{s_7, s_4\}, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1), k_2), k_1), k_1)$$
$$= \{\{s_3, s_6\}, E(\{s_7, s_4\}, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1), k_2)\}$$

Then c_1 sends the decrypted path to s_1.
After passing through s_3, the packet reaches to s_6. Where s_6 is an edge switch it sends $E(\{s_7, s_4\}, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1), k_2)$ to c_2, then c_2 decrypts it using k_2.

$$D(E(\{s_7, s_4\}, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1), k_2), k_2)$$
$$= \{\{s_7, s_4\}, E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1)\}$$

After passing through s_7, the packet reaches to s_4. Where s_4 is an edge switch it sends $E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1)$ to c_1, then c_1 decrypts it using k_1.

$$D(E(\{s_8, E(\{s_9, s_{10}\}, k_3)\}, k_1), k_1)$$
$$= \{s_8, E(\{s_9, s_{10}\}, k_3)\}$$

The packet reaches to s_8. Where s_8 is an edge switch it sends $E(\{s_9, s_{10}\}, k_3)$ to c_3, then c_3 decrypts it using k_3.

$$D(E(\{s_9, s_{10}\}, k_3), k_3)$$
$$= \{s_9, s_{10}\}$$

After passing through s_9, the packet finally reaches to destination s_{10}.

5. CONCLUSION

In this paper, we introduce the attacks that are possible in a source routing multi domain SDN. We proposed a novel encryption and decomposition techniques to find out the encrypted shortest path between the switches, which eventually hides the topology of one domain to another. We also introduce threshold value at which the secret keys shared between the master and slave controllers are to be recomputed, to prevent the attacker from deriving the network topology.
The first attempt is made in this paper for hiding the network topology in a source routing multi domain SDN, which is highly significant in a multi tenant cloud environment. The proposed approach can effectively countermeasure any kind of attacks which reveals the network topology.

6. REFERENCES

[1] Marc Mendonca, Bruno Astuto A. Nunes, Xuan-Nam Nguyen, Katia Obraczka and Thierry Turletti, *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks*. hal-00825087, version 2, 2013.

[2] Ramya Raghavendra, Jorge Lobo and Kang-Won Lee, *Dynamic Graph Query Primitives for SDN-based Cloud Network Management*. In Proceedings of HotSDN, 2012.

[3] Seungwon Shin, Guofei Gu, *Attacking Software-Defined Networks: A First Feasibility Study*. HotSDN13, August 16, 2013.

[4] Diego Kreutz, Fernando M. V. Ramos,Paulo Verissimo, *Towards Secure and Dependable Software-Defined Networks*. HotSDN13, August 16, 2013.

[5] Xitao Wen,Yan Chen, Chengchen Hu, *Towards a Secure Controller Platform for OpenFlow Applications*. HotSDN13, August 16, 2013.

[6] Stefan Schmid, Jukka Suomela *Exploiting Locality in Distributed SDN Control*. HotSDN13, August 16, 2013.

[7] Alistair Moffat and Tadao Takaoka, *An all pairs shortest path algorithm with expexted running time* $O(n^2 \log n)$. IEEE, 1985.

[8] Bloniarz P.A., *A shortest path algorithm with expected time* $O(n^2 \log n \log n)$. SIAM J. Comput.,12(1983), 588-600

[9] S. Chaudhuri and C. D. Zaroliagis, *Shortest Paths in Digraphs of Small Treewidth. Part I: Sequential Algorithms*. Algorithmica, 27:212226, 2000.

[10] C. Demetrescu and G. F. Italiano, *Experimental analysis of dynamic all pairs shortest path algorithms*. ACM Trans.Algorithms, 2(4), Oct. 2006.

[11] E. W. Dijkstra, *A note on two problems in connexion with graphs*. NUMERISCHE MATHEMATIK, 1(1):269271,1959.

[12] F. Wei, *TEDI: efficient shortest path query answering on graphs*. In Proceedings of SIGMOD, 2010.

[13] Marc Mendonca, Bruno Astuto A. Nunes, Xuan-Nam Nguyen, Katia Obraczka and Thierry Turletti, *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks*. hal-00825087, version 2, 2013.