Recent Trends in Intensive Computing
M. Rajesh et al. (Eds.)

doi:10.3233/APC210224

Shielding Software Defined Network Using Hidden Authentication Technique

B. Srivani [a] [1], S. Renu Deepti [a], and Qualid Unnisa [a]
[a]*Department of Information Technology, VNR VJIET, Hyderabad, India*

Abstract. Software defined networking (SDN) permits community feature program ability intended to facilitate about design along with renovation, as well as permit community directors toward adapt congestion guidelines. Nevertheless, denial of provider (DoS) assailants causes productivity issues upon centralized consolidate aircraft about SDN. Even through shipping layer safety (TLS) be able to assist comfy manage plane, that far analytically extensive and composite design. Within the document, we plan light-weight validate compound, known as Hidden Authentication (HiAuth), toward guard the SDN through battering specifications about redirecting devices to control packets thru effective bitwise functioning. HiAuth be that initially toward incorporate records battering methods for Open Flow toward offer safety in opposition to DoS attacks. HiAuth utilizes IP identification field about IPv4 as well as proceedings recognition area about OpenFlow within two attestation methods. The investigational outcomes display that HiAuth able to efficiently alleviate trespasser DoS assaults as well as supply excessive unnoticeable toward assailants.

Keywords. Software defined networking (SDN), OpenFlow, Denial of service (DoS) attacks, Information hiding

1. Introduction

Software defined networking (SDN) disassociate the system manipulate through redirecting gadgets as a result simplifies and complements community maintenance [1]. Within fashionable, SDN shape contains three layers: software plane, manage plane, and data aircraft. Structure and the attachments a few of the three layers abouts SDN. The northbound utility programming interfaces (APIs) permit SDN packages toward grant plant as well as protection guidelines to command aircraft. Later, SDN manipulate aircraft enforce one's regulations at forward gadgets through the southbound APIs. OpenFlow is the first SDN popular which defines open southbound interfaces for controlling network flows. The SDN centralization of community intelligence with the supply about worldwide view of the whole community improved programmability and scalability for destiny network and service manage. As an end result, many new SDN packages are proposed to optimize the network performance from incredible factors, which includes throughput maximization [2][3], deterministic delay functionality to influence the network behavior via software program from a logically centralized manage brings numerous benefits.

However, software vulnerabilities grow to be a project [7]. More critically, the centralization of the control aircraft can motive an unmarried aspect of nonsuccess.

[1] B. Srivani,Department of Information Technology, VNR VJIET, Hyderabad
Email: srivaanib@gmail.com.

Regrettably, transport layer protection (TLS) be optionally available with OpenFlow protocol.

Because of its configuration complexity, TLS isn't adopted by means of way of a few OpenFlow-enabled switches and controllers [8]. As a stop result, the legitimacy of the forwarding gadgets can't be verified. Hence, malicious assults be able to without issues release denial of provider (DoS) attacks closer to SDN controller. Toward solve the DoS problem, we generalize danger worries about DoS attacks towards the SDN controller. In unique, we observe feasible DoS assault eventualities, which include the outsider and the insider attack lessons. Then, we compare DoS affects at the SDN controller. Eventually, designed a light-weight attestation scheme, Hidden Authentication (HiAuth), toward reduce attatrespasser DoS assults. HiAuth have essential capabilities: 1) difficult to understand, 2) light-weight. Initially, intended to obscurity, HiAuth mimics the original statistical distributions of the values cause through the going for walks frameworks towad say unpredicted through attackers. Then having mild-weight, HiAuth simplest is based totally on easy bitwise operations for computations.

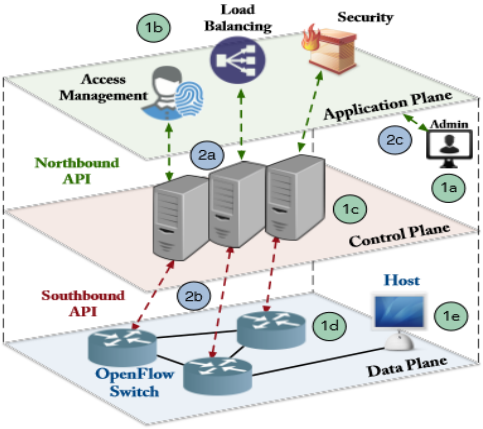


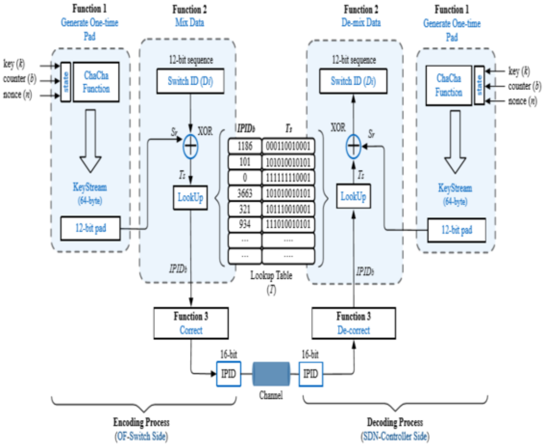
Figure 1. SDN architecture and its security challenges hiding

Thus, it does no longer require specialized hardware. Toward excellent about information, HiAuth be initiate toward consist of records battering models for OpenFlow rule toward gives safety in case of DoS assults on rule degree. There, recommend two HiAuth schemes: IP identification (IPID) based totally definitely HiAuth hids recognization about redirecting things inside IPID province about the IPv4 header. Transaction identification (XID) base absolutely HiAuth what hide the recognization about the redirecting things of XID area about OpenFlow header. IPID-primarily base completely HiAuth presents attestation on the community layer. Nevertheless, within IPv4 turns into out of date in the furtherer else total change toward IPv6 takes place, XIDbased HiAuth may use toward equal stage of attestation.

2. Research Methodology

SDN has two major blessings: community application potential and centralized community control. Initially, through seperate about manage plane as well as statistics plane, program of SDN lets in system rules toward changed with the aid about software program rather the guide composition correlated toward conventional systems. Toward renovate community model in conventional systems, every device must be manually configured, which may additionally reason as a result toward safety susceptibility. Next, the consolidated manipulate common sense able to ease system maintenance due to supply about system worldwide view. Protectivity within SDN have aspect, particularly security via SDN. On the only hand, safety thru SDN makes a specialty of utilizing SDN features to solve conventional community protection problems. SDN protectivity packages able to look at packets via manipulate aircraft. Then protectivity analysis, those application be able to drop or else redirected traffic toward safety center boxes. Upon alternative, protection about SDN offers through safety challenges resulting from SDN, e.G., factor about failure. To similarly difficult the differences among these elements, we took DoS elimination for instance. With safety over SDN, DoS assault upon classical systems may prevent through SDN applications.

Defense4All plays two important responsibilities, particularly (a) conduct observing through using studying traffic records, (b) traffic redirected through the server underneath DoS assault inside community can be covered through such SDN programs. Within case safety about SDN, DoS attack in opposition to manipulate plane may be prevented. TLS charge methods be the solution encouraged by using the current SDN requirements. The planned HiAuth procedure belongs toward the latter issue wherein targets toward a lightweight packet stage attestation toward save you DoS assaults towards manipulate aircraft.



$IPID_b$	T_s
1186	000110010001
101	101010010101
0	111111110001
3663	101010010101
321	101110010001
934	111010010101
...	....
...	....

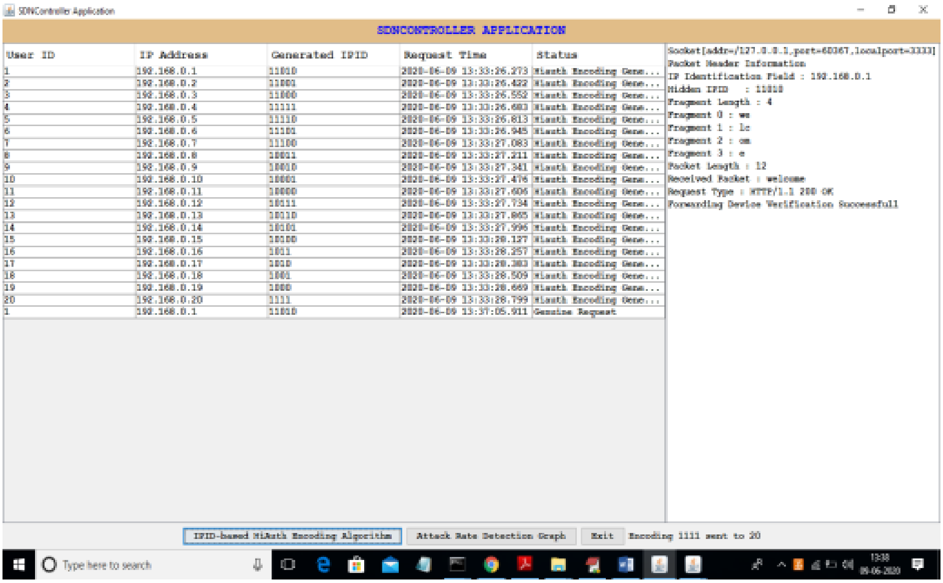
Figure 2. IPID-based HiAuth

HiAuth be packet-degree attestation procedure focus toward mitigate elegance about external DoS assaults within opposition to manage aircraft. The most effective required a bitwise operation and a easy mapping characteristic to cover identification facts about gadget for control packet header. Upon opposite hand, HiAuth design be instead honest and easy at the same time as compare toward TLS layout. Within phase, we introduced two HiAuth methods specifically, IPID-primarily base HiAuth along with XIDbased HiAuth.

IPID-base totally HiAuth be plan toward used for inside systems so as to be put into effect chance of IPID generation. In spite of paintings posted inside the writing upon hide records savvy IPID, it far cleans so to be more about these strategies forget apparent adjustment what to be brought the issuing about IPID. Moreover, the total techniques be useful best whilst community MTU be set on. Consequently, suggest IPID-based HiAuth toward offer data attestation as well as triumph over the deficiency. The IPID-primarily base HiAuth encoding along with deciphering methods be regularity along contains 3 consequences: (1) one-time pad era, (2) information mining, (3) distribution corrective model. HiAuth encoding technique calls for subsequent inputs: a) a 256-bit key, a sixty-four-bit block counter, and a 64bit for use in the one-time pad generator, b) a 12-bit tool recognition for use within the information blending characteristic. The output of those abilities be the mapped right into 12-bit IPID base cost. Eventually, a distribution corrective model changes the IPID base cost right to issue compliant sixteen-bit IPID if you want to be used for the manage packet transmission.

3. Results and Analysis

In this section, we demonstrate the results of our computations, which gain the security by hiding its identity of the network without leaving the traces behind as per the proposed technique, hence it mitigates the intruder. Also, it describes and demonstrates the computational analysis precisely with minimal efforts and good security comparatively.


SDNController Application
SDNCONTROLLER APPLICATION
IPID-based HiAuth Encoding Algorithm
Attack Rate Detection Graph
Exit
Type here to search

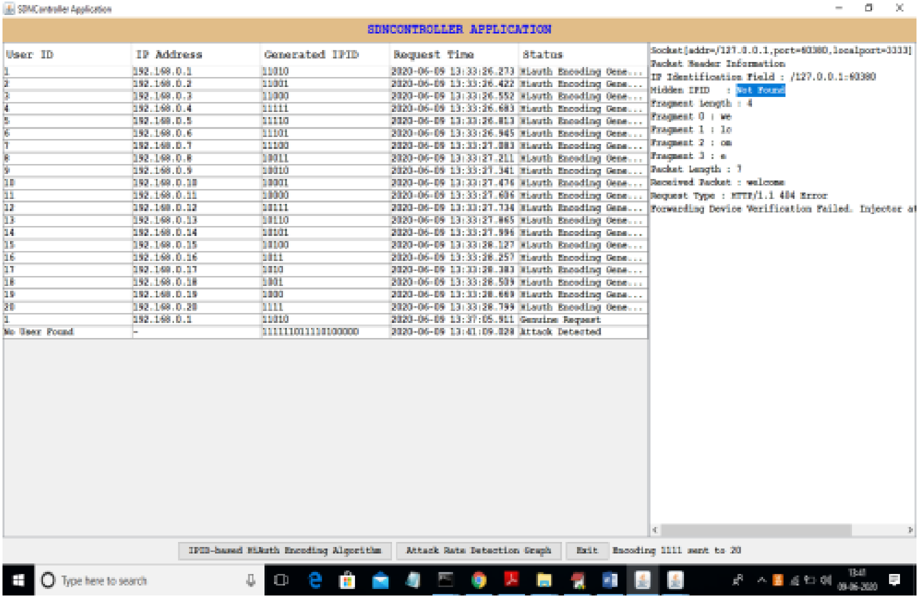
User ID	IP Address	Generated IPID	Request Time	Status
1	192.168.0.1	11010	2020-06-09 13:33:26.273	HiAuth Encoding Gene...
2	192.168.0.2	11001	2020-06-09 13:33:26.422	HiAuth Encoding Gene...
3	192.168.0.3	11000	2020-06-09 13:33:26.552	HiAuth Encoding Gene...
4	192.168.0.4	11111	2020-06-09 13:33:26.683	HiAuth Encoding Gene...
5	192.168.0.5	11110	2020-06-09 13:33:26.813	HiAuth Encoding Gene...
6	192.168.0.6	11101	2020-06-09 13:33:26.945	HiAuth Encoding Gene...
7	192.168.0.7	11100	2020-06-09 13:33:27.083	HiAuth Encoding Gene...
8	192.168.0.8	10011	2020-06-09 13:33:27.211	HiAuth Encoding Gene...
9	192.168.0.9	10010	2020-06-09 13:33:27.341	HiAuth Encoding Gene...
10	192.168.0.10	10001	2020-06-09 13:33:27.476	HiAuth Encoding Gene...
11	192.168.0.11	10000	2020-06-09 13:33:27.606	HiAuth Encoding Gene...
12	192.168.0.12	10111	2020-06-09 13:33:27.734	HiAuth Encoding Gene...
13	192.168.0.13	10110	2020-06-09 13:33:27.865	HiAuth Encoding Gene...
14	192.168.0.14	10101	2020-06-09 13:33:27.996	HiAuth Encoding Gene...
15	192.168.0.15	10100	2020-06-09 13:33:28.127	HiAuth Encoding Gene...
16	192.168.0.16	1011	2020-06-09 13:33:28.257	HiAuth Encoding Gene...
17	192.168.0.17	1010	2020-06-09 13:33:28.383	HiAuth Encoding Gene...
18	192.168.0.18	1001	2020-06-09 13:33:28.509	HiAuth Encoding Gene...
19	192.168.0.19	1000	2020-06-09 13:33:28.669	HiAuth Encoding Gene...
20	192.168.0.20	1111	2020-06-09 13:33:28.799	HiAuth Encoding Gene...
1	192.168.0.1	11010	2020-06-09 13:37:05.911	Genuine Request

```
Socket[addr=/127.0.0.1,port=60367,localport=3333]
Packet Header Information
IP Identification Field : 192.168.0.1
Hidden IPID   : 11010
Fragment Length : 4
Fragment 0 : we
Fragment 1 : lc
Fragment 2 : om
Fragment 3 : e
Packet Length : 12
Received Packet : welcome
Request Type : HTTP/1.1 200 OK
Forwarding Device Verification Successfull
```

Encoding 1111 sent to 20

Figure 3. Hiding the IPID of the request

In Figure 3 we can see that the received request in text area has all the details of its header, wrapped with the hidden authentication mechanism with all the request details in text area we can see number of fragments received and data in each fragment and hidden IPID and then verification message.


SDNController Application
SDNCONTROLLER APPLICATION
IPID-based HiAuth Encoding Algorithm
Attack Rate Detection Graph
Exit
Type here to search
13:41
09-06-2020

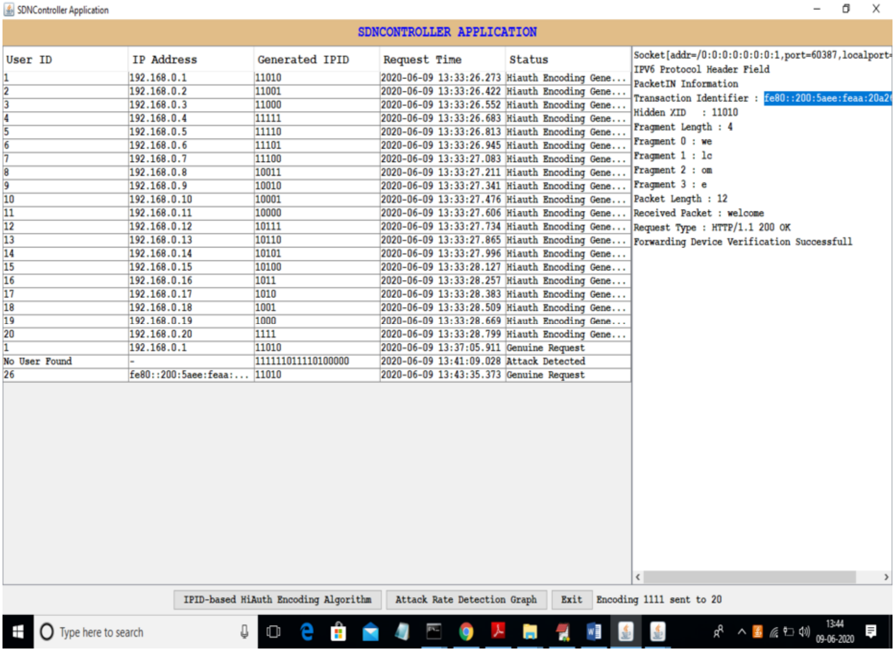
User ID	IP Address	Generated IPID	Request Time	Status
1	192.168.0.1	11010	2020-06-09 13:33:26.273	Hiauth Encoding Gene...
2	192.168.0.2	11001	2020-06-09 13:33:26.422	Hiauth Encoding Gene...
3	192.168.0.3	11000	2020-06-09 13:33:26.552	Hiauth Encoding Gene...
4	192.168.0.4	11111	2020-06-09 13:33:26.683	Hiauth Encoding Gene...
5	192.168.0.5	11110	2020-06-09 13:33:26.813	Hiauth Encoding Gene...
6	192.168.0.6	11101	2020-06-09 13:33:26.945	Hiauth Encoding Gene...
7	192.168.0.7	11100	2020-06-09 13:33:27.083	Hiauth Encoding Gene...
8	192.168.0.8	10011	2020-06-09 13:33:27.211	Hiauth Encoding Gene...
9	192.168.0.9	10010	2020-06-09 13:33:27.341	Hiauth Encoding Gene...
10	192.168.0.10	10001	2020-06-09 13:33:27.476	Hiauth Encoding Gene...
11	192.168.0.11	10000	2020-06-09 13:33:27.606	Hiauth Encoding Gene...
12	192.168.0.12	10111	2020-06-09 13:33:27.734	Hiauth Encoding Gene...
13	192.168.0.13	10110	2020-06-09 13:33:27.865	Hiauth Encoding Gene...
14	192.168.0.14	10101	2020-06-09 13:33:27.996	Hiauth Encoding Gene...
15	192.168.0.15	10100	2020-06-09 13:33:28.127	Hiauth Encoding Gene...
16	192.168.0.16	1011	2020-06-09 13:33:28.257	Hiauth Encoding Gene...
17	192.168.0.17	1010	2020-06-09 13:33:28.383	Hiauth Encoding Gene...
18	192.168.0.18	1001	2020-06-09 13:33:28.509	Hiauth Encoding Gene...
19	192.168.0.19	1000	2020-06-09 13:33:28.669	Hiauth Encoding Gene...
20	192.168.0.20	1111	2020-06-09 13:33:28.799	Hiauth Encoding Gene...
1	192.168.0.1	11010	2020-06-09 13:37:05.911	Genuine Request
No User Found	-	111111011110100000	2020-06-09 13:41:09.028	Attack Detected

```
Socket[addr=/127.0.0.1,port=60380,localport=3333]
Packet Header Information
IP Identification Field : /127.0.0.1:60380
Hidden IPID    : Not Found
Fragment Length : 4
Fragment 0 : we
Fragment 1 : lc
Fragment 2 : om
Fragment 3 : e
Packet Length : 7
Received Packet : welcome
Request Type : HTTP/1.1 404 Error
Forwarding Device Verification Failed. Injector at
```

Encoding 1111 sent to 20

Figure 4. Rejecting the unsecure request

In Figure 4 the controller did not find the ID to be secured in the text area and hence rejected the request as the verification failed.


SDNController Application
SDNCONTROLLER APPLICATION
IPID-based HiAuth Encoding Algorithm
Attack Rate Detection Graph
Exit
Type here to search
13:44
09-06-2020

User ID	IP Address	Generated IPID	Request Time	Status
1	192.168.0.1	11010	2020-06-09 13:33:26.273	Hiauth Encoding Gene...
2	192.168.0.2	11001	2020-06-09 13:33:26.422	Hiauth Encoding Gene...
3	192.168.0.3	11000	2020-06-09 13:33:26.552	Hiauth Encoding Gene...
4	192.168.0.4	11111	2020-06-09 13:33:26.683	Hiauth Encoding Gene...
5	192.168.0.5	11110	2020-06-09 13:33:26.813	Hiauth Encoding Gene...
6	192.168.0.6	11101	2020-06-09 13:33:26.945	Hiauth Encoding Gene...
7	192.168.0.7	11100	2020-06-09 13:33:27.083	Hiauth Encoding Gene...
8	192.168.0.8	10011	2020-06-09 13:33:27.211	Hiauth Encoding Gene...
9	192.168.0.9	10010	2020-06-09 13:33:27.341	Hiauth Encoding Gene...
10	192.168.0.10	10001	2020-06-09 13:33:27.476	Hiauth Encoding Gene...
11	192.168.0.11	10000	2020-06-09 13:33:27.606	Hiauth Encoding Gene...
12	192.168.0.12	10111	2020-06-09 13:33:27.734	Hiauth Encoding Gene...
13	192.168.0.13	10110	2020-06-09 13:33:27.865	Hiauth Encoding Gene...
14	192.168.0.14	10101	2020-06-09 13:33:27.996	Hiauth Encoding Gene...
15	192.168.0.15	10100	2020-06-09 13:33:28.127	Hiauth Encoding Gene...
16	192.168.0.16	1011	2020-06-09 13:33:28.257	Hiauth Encoding Gene...
17	192.168.0.17	1010	2020-06-09 13:33:28.383	Hiauth Encoding Gene...
18	192.168.0.18	1001	2020-06-09 13:33:28.509	Hiauth Encoding Gene...
19	192.168.0.19	1000	2020-06-09 13:33:28.669	Hiauth Encoding Gene...
20	192.168.0.20	1111	2020-06-09 13:33:28.799	Hiauth Encoding Gene...
1	192.168.0.1	11010	2020-06-09 13:37:05.911	Genuine Request
No User Found	-	111111011110100000	2020-06-09 13:41:09.028	Attack Detected
26	fe80::200:5aee:feaa:...	11010	2020-06-09 13:43:35.373	Genuine Request

```
Socket[addr=/0:0:0:0:0:0:0:1,port=60387,localport=
IPV6 Protocol Header Field
PacketIN Information
Transaction Identifier : fe80::200:5aee:feaa:20a2
Hidden XID    : 11010
Fragment Length : 4
Fragment 0 : we
Fragment 1 : lc
Fragment 2 : om
Fragment 3 : e
Packet Length : 12
Received Packet : welcome
Request Type : HTTP/1.1 200 OK
Forwarding Device Verification Successfull
```

Encoding 1111 sent to 20

Figure 5. Approving the authenticated request

In Figure 5 the text area this time we can see at selected text that this request process as IPV6 address and if SDN found correct XID then device request will be authenticated. If request not authenticated then user not found in the look table or not authenticated with XID process and display information as Attack detected.

4. Conclusion

In this research paper, we introduce packet authentications concept called IPID and XID. IPID used IPV4 protocol to hide device id in packets along with data and then send packet to SDN controller where IPID will get authenticated by extracting details from packet. Device ID will be encrypted using CHA-CHA algorithm by generating random number and then perform XOR operation between CHA-CHA random number and device id to get secure device ID with the help of DATA MIXING. Secure device id will get exchange between genuine device and SDN controller and get authenticated for each request. After assigning secure ID SDN controller will used lookup table to check whether forwarding device id exists or not. If exists device will be considered as genuine else malicious. In future if protocol changed from IPV4 to IPV6 then IPID packet fragmentation will not work so author using XID transaction-based header to hide device id. XID will hide device details in transaction header and to ease computation look up table will not be used and XID will dynamically compute device ID for authentication upon each request.

References

[1] M. Huang, W. Liang, Z. Xu, and S. Guo, “Efficient algorithms for throughput maximization in software-defined networks with consolidated middleboxes,” IEEE Transactions on Network and Service Management, vol. 14, no. 3, pp. 631–645, 2017.

[2] Y.-J. Chen, L.-C. Wang, F.-Y. Lin, and B.-S. P. Lin, “Deterministic quality of service guarantee for dynamic service chaining in software defined networking,” IEEE Transactions on Network and Service Management, vol. 14, no. 4, pp. 991–1002, 2017.

[3] H. Hamed and E. Al-Shaer, “Taxonomy of conflicts in network security policies,” IEEE Communications Magazine, vol. 44, no. 3, pp. 134–141, 2006.

[4] S. J. Vaughan-Nichols, “OpenFlow: The next generation of the network?,” IEEE Computer, vol. 44, no. 8, pp. 13–15, 2011.

[5] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, “Security in software defined networks: A survey,” IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2317–2346, 2015.Yin B., Shen W., Cheng Y., Cai L.X. and Li Q, “Distributed resource sharing in fog-assisted big data streaming”, IEEE international conference on communications (ICC), pp. 1-6, May 2017.

[6] W. Mazurczyk and L. Caviglione, “Steganography in modern smartphones and mitigation techniques,” IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 334–357, 2015.

[7] J.-M. Guo, G.-H. Lai, K. Wong, and L.-C. Chang, “Progressive halftone watermarking using multi layer table lookup strategy,” IEEE Transactions on Image Processing, vol. 24, no. 7, pp. 2009–2024, 2015

[8] B. Jankowski, W. Mazurczyk, and K. Szczypiorski, “Padsteg: Introducing inter-protocol steganography,” Telecommunication Systems, vol. 52, no. 2, pp. 1101–1111, 2013.