© 2018 IJSRST | Volume 4 | Issue 5 | Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Science and Technology
doi : https://doi.org/10.32628/IJSRST

Enhancing IoT Network Security Using SDN Controller and Blockchain Technology

[1]Prof. Piyush K Ingole, [2]Prof. Apeksha V. Sakhare
[1,2]G.H.Raisoni COE, Nagpur, India

ABSTRACT

To beautify the security of SDN network In order to enhance the security of SDN (Software-defined networking) networks utilized within cloud environments, SDN has been upgraded, thus altering the standard procedure of the existing network. This study aims to implement an SDN-approved blockchain application on the cloud. Likely, the SDN controller ryu is employed for network management and customization. This review presents an overview of common security issues encountered with SDN when integrated with cloud environments. It introduces emerging ideas in the newly launched blockchain model and proposes reasons that support blockchain as a formidable security feature when solutions are deployed where SDN and cloud are interconnected. Given the significant increase in the amount of user data (private, enterprise, commercial, etc.) available online, this puts them at high risk from malicious users. Various security measures have been proposed and implemented to protect user data against undefined risks. Many of these solutions utilize traditional networking approaches that are intricate and challenging to manage. These approaches rely on the manual configuration of devices, causing policy conflicts that may compromise network security.

Keywords : SDN, Software-defined networking

Introduction

Software Defined Networking (SDN) is the framework for network architectures that separates control plane from forwarding plane making the network management more truthful. The manipulate usual insight of network is completed by a reasonably centralised network controller building switching & routing gadgets as simple information forwarding devices. Famous agencies like Microsoft, Google, Yahoo, Facebook, and Verizon have invested within the improvement of open standards for SDN. OpenFlow protocol is the open requirements that allows communication among the manage plane and the records plane in SDN environment[1]. The controller uses OpenFlow protocol to bypass switching, routing, load balancing or firewall policies onto information plane devices.

Firewall can be conceptualized as a security tool build totally on prescribed safety pointers utilized for tracing and managing outward & inward packet site traffic in a network. A traditional firewall serves like a shield amongst an internal trusted community & an external suspected communnity alongwith the internet[2]. Because of the reality the centralised manage in SDN endorse the imposition of network-big security guidelines & intercept insurance conflicts, it is extremely pleasant to implement firewall with SDN network structure[3].

In this theory, a firewall safety configuration is recommended. It is intended to offer network-substantial security at identical time as analysing inward stream to the community. This solution offers the network administrator total manipulate over safety coverage execution & modification; simultaeneously building the firewall authentication in the direction of threats through monitoring community stream.

II. BLOCK CHAIN

A blockchain may be a decentralised, dispensed & publicized virtual ledger. It sincerely wants to report logs throughout various computer networks such that some concerned document can not be modified retrospectively, without the modification of each next block. It allows the personnel to validate and scrutinize transactions discretely and comparatively economically[4]. A p2p network & allocated timestamping server can control a blockchain statistics independently. They are unfeigned by using mass cooperation fascinated through common self-hobbies. This method allows strong paintings go with the flow wherever contributors' unpredictability regarding statistics protection is insignificant. The feature of countless reliableness from a virtual plus can be removed by using blockchain. It verifies that every precious unit became transmittted once, locating the extensive snag of dual dispensation. Blockchain is defined as a price-trade protocol. Blockchain holds call authority due to the fact of once well developed to detail the trade agreement[5]. It gives a file which impels offer and attraction.

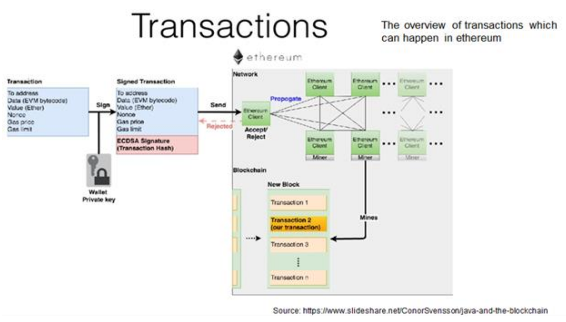


Fig. 1. Blockchain Transactions

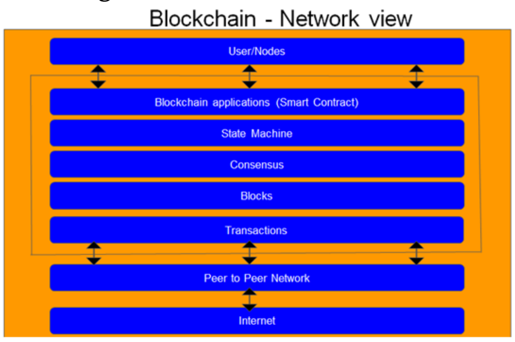


Fig. 2. Network view of Blockchain

A. Ethereum Blockchain

1)Conceived in November-2013 by Vitalik Buterin.
2)Turing Complete Language
3) Permits advancement of random programs (smart contracts)
4) Basic launch as“Frontier”, the leading-edge launch as “homestate launch” & final launch i.e. “serenity”

5)Serenityproof of Stake (Caster), Adaptability, Secrecy,EVM.
6)The visionary and prophetic of Web 3.0 idea – Public, applications, details and internet linked together.
7)DNS, Search Engines & identity at the internet – distributed in net 3.0 and Ethereum to understand it.
8) Native foreign money ether ETH- difficult forked model.

The blockchain comprises of triple component technique running in combination i.e. cryptographic analysis, uneven public key cryptography, dispensed P2P computing.

i)A root-analysis of a complete chain along with the analysis of transactions inside the block comprises each block header.

ii) An encryption seed is created by using the bits of facts within the block header, that in turn creates a DAG file, that extends to 1GB and functions like a sort of father-up component array for proof-of-work rule set that smashes collectively slab of data from DAG so as to point towards a triumphing nonce rate so that you can authenticate the slab.

iii)A pair of cryptographic keys, one general & one personal are used by ethereum debts to encode transactions dispatched to their concerned virtual devices. Array of pointers applied is recognized to be secp256k1 curve to perform encoding.

iv)Ethereum uses the elliptic-curve build completely on encryption protocol called as ECDSA algorithm allows for short key extent that lowers stowage goals & transferral specifications.

V) SHA-256 is used by ethereum

III. MININET IOT NETWORK

Mininet-IoT emulates a complete network of nodes and links on a single machine. To create a sample two-host, one-switch community, simply run:

```
sudo mn
```

For establishing vast network at confined assets of a smooth lone computer or virtual machine, mininet is used as an emulator. For assisting exploration in software defined networking (SDN) and openflow, mininet is designed. Mininet simulator permits running unvaried codes collectively on virtual machine on easy computer. It offers consolation and authenticity at extremely economical cost. The possibility to mininet is hardware check beds that are quick, precise yet extremely costly & shared. Another option is using simulator which is too economical but every so often gradual and need code moderation. Mininet provides user friendliness, overall functioning precision & adaptability.

IV. IMPLEMENTATION

A. Ethereum Libraries (to be installed in python)

1.npm
2.ganache-cli
3.web-3 python
4.flask
- flask – restful
- flask – marshmallow
5.install solidity
6.install ethereum
7.install solc

8.py-solc

9.py-solc-x

B. Ethereum Implementation

Access key has to be generated from infura.io website to generate Eth wallet. This website is used to access ethereum. A new project option will be displayed on website. Project secret key will be given to create new project.

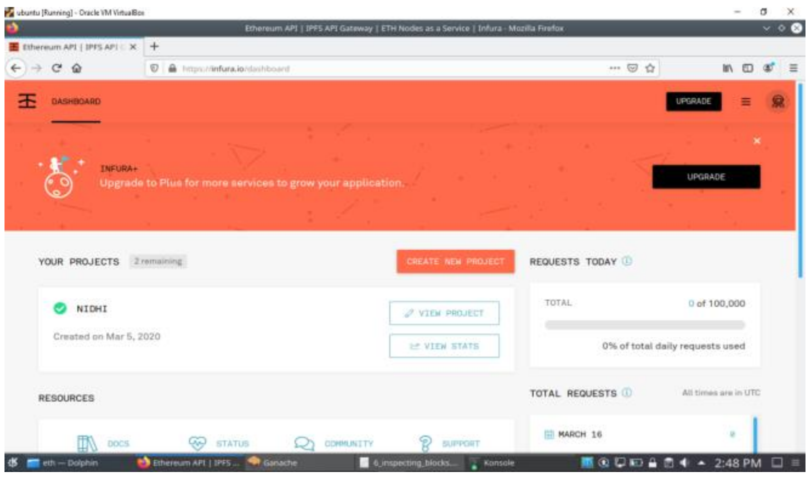
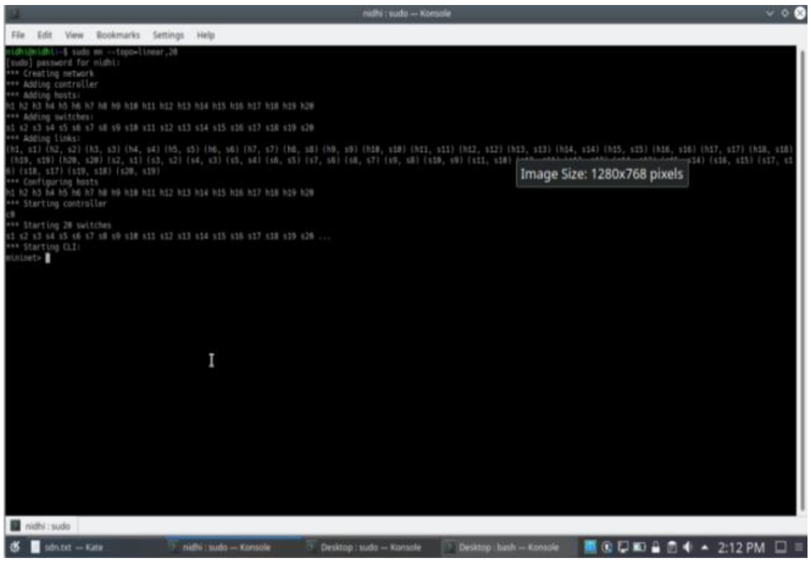

ubuntu [Running] - Oracle VM VirtualBox
Ethereum API | IPFS API Gateway | ETH Nodes as a Service | Infura - Mozilla Firefox
https://infura.io/dashboard
DASHBOARD
UPGRADE
INFURA+
Upgrade to Plus for more services to grow your application.
UPGRADE
YOUR PROJECTS
2 remaining
CREATE NEW PROJECT
REQUESTS TODAY
NIDHI
Created on Mar 5, 2020
VIEW PROJECT
VIEW STATS
TOTAL
0 of 100,000
0% of total daily requests used
RESOURCES
TOTAL REQUESTS
All times are in UTC
DOCS
STATUS
COMMUNITY
SUPPORT
MARCH 16
2:48 PM

Fig. 3.infura.io website

Ganache needs to be downloaded and installed. It is locally deployed in ethereum. Ganache screen will be displayed where outputs regarding transactions, records, accounts etc.are displayed.

A various end-hosts, switches, routers, links on a Linux can be simulated by using mininet. It is an open source network emulation environment. Mininet has to be started by following code:

sudo mn –topo-linear.20

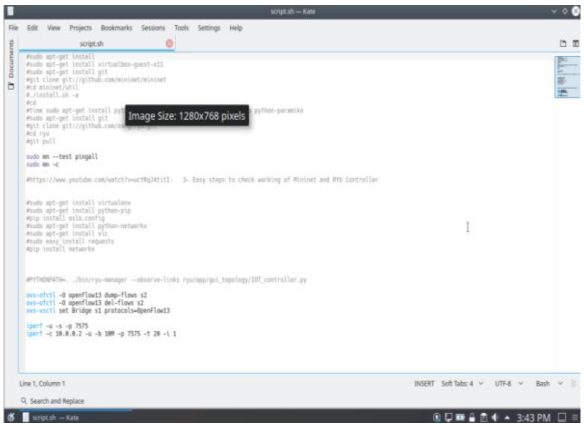


```
nidhi@nidhi:~$ sudo mn --topo=linear,20
[sudo] password for nidhi:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (h7, s7) (h8, s8) (h9, s9) (h10, s10) (h11, s11) (h12, s12) (h13, s13) (h14, s14) (h15, s15) (h16, s16) (h17, s17) (h18, s18)
 (h19, s19) (h20, s20) (s2, s1) (s3, s2) (s4, s3) (s5, s4) (s6, s5) (s7, s6) (s8, s7) (s9, s8) (s10, s9) (s11, s10) [illegible] s14) (s16, s15) (s17, s1
6) (s18, s17) (s19, s18) (s20, s19)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
*** Starting controller
c0
*** Starting 20 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 ...
*** Starting CLI:
mininet>
```

nidhi : sudo — Konsole
File Edit View Bookmarks Settings Help
Image Size: 1280x768 pixels
2:12 PM

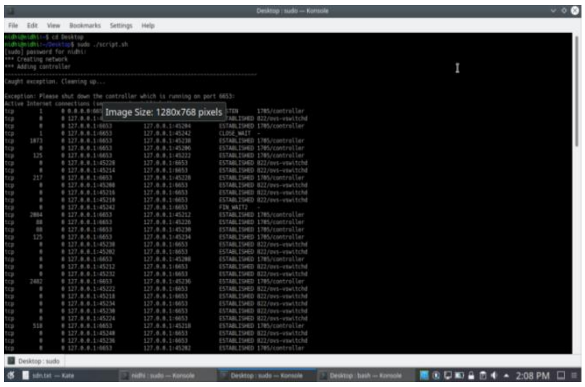
Fig. 4. Mininet start

Open v-switch (Open flow) to access following code.


script.sh — Kate
File Edit View Projects Bookmarks Sessions Tools Settings Help
script.sh
Documents
Image Size: 1280x768 pixels
Line 1, Column 1
INSERT Soft Tabs: 4 UTF-8 Bash
Search and Replace
script.sh — Kate
3:43 PM

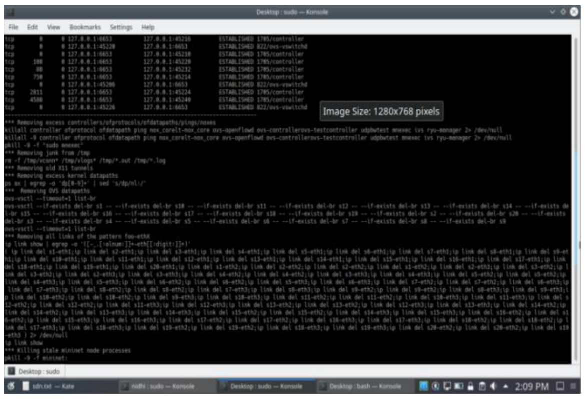
```
#sudo apt-get install
#sudo apt-get install virtualbox-guest-x11
#sudo apt-get install git
#git clone git://github.com/mininet/mininet
#cd mininet/util
#./install.sh -a
#cd
#time sudo apt-get install py[illegible] python-paramiko
#sudo apt-get install git
#git clone git://github.com/[illegible]
#cd ryu
#git pull

sudo mn --test pingall
sudo mn -c

#https://www.youtube.com/watch?v=ucYRqJ4tit1   3- Easy steps to check working of Mininet and RYU Controller


#sudo apt-get install virtualenv
#sudo apt-get install python-pip
#pip install oslo.config
#sudo apt-get install python-networkx
#sudo apt-get install vlc
#sudo easy_install requests
#pip install networkx



#PYTHONPATH=. ./bin/ryu-manager --observe-links ryu/app/gui_topology/IOT_controller.py

ovs-ofctl -O openflow13 dump-flows s2
ovs-ofctl -O openflow13 del-flows s2
ovs-vsctl set Bridge s1 protocols=OpenFlow13

iperf -u -s -p 7575
iperf -c 10.0.0.2 -u -b 10M -p 7575 -t 20 -i 1
```

Fig. 5a. Open v-switch code



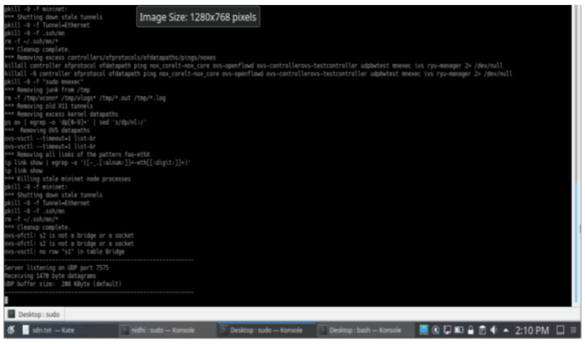
```
nidhi@nidhi:~$ cd Desktop
nidhi@nidhi:~/Desktop$ sudo ./script.sh
[sudo] password for nidhi:
*** Creating network
*** Adding controller
------------------------------------------------------------------------------------
Caught exception. Cleaning up...

Exception: Please shut down the controller which is running on port 6653:
Active Internet connections (se[illegible]
tcp        1      0 0.0.0.0:665[illegible]                          [illegible]TEN      1705/controller
tcp        0      0 127.0.0.1:[illegible]                           [illegible]TABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:6653          127.0.0.1:45204         ESTABLISHED 1705/controller
tcp        1      0 127.0.0.1:6653          127.0.0.1:45242         CLOSE_WAIT  -
tcp     1873      0 127.0.0.1:6653          127.0.0.1:45238         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:6653          127.0.0.1:45206         ESTABLISHED 1705/controller
tcp      125      0 127.0.0.1:6653          127.0.0.1:45222         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:45228         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45214         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp      217      0 127.0.0.1:6653          127.0.0.1:45228         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:45200         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45216         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45210         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45242         127.0.0.1:6653          FIN_WAIT2   -
tcp     2084      0 127.0.0.1:6653          127.0.0.1:45212         ESTABLISHED 1705/controller
tcp       88      0 127.0.0.1:6653          127.0.0.1:45226         ESTABLISHED 1705/controller
tcp       88      0 127.0.0.1:6653          127.0.0.1:45230         ESTABLISHED 1705/controller
tcp      125      0 127.0.0.1:6653          127.0.0.1:45234         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:45238         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45202         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:6653          127.0.0.1:45200         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:45212         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45232         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp     2482      0 127.0.0.1:6653          127.0.0.1:45236         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:45222         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45218         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45234         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45230         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45224         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp      518      0 127.0.0.1:6653          127.0.0.1:45218         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:45240         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:45236         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:6653          127.0.0.1:45202         ESTABLISHED 1705/controller
```

Fig. 5b. Output of open v-switch



```
tcp        0      0 127.0.0.1:6653          127.0.0.1:45216         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:45220         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp        0      0 127.0.0.1:6653          127.0.0.1:45210         ESTABLISHED 1705/controller
tcp      100      0 127.0.0.1:6653          127.0.0.1:45220         ESTABLISHED 1705/controller
tcp       88      0 127.0.0.1:6653          127.0.0.1:45232         ESTABLISHED 1705/controller
tcp      750      0 127.0.0.1:6653          127.0.0.1:45214         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:45206         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
tcp     2811      0 127.0.0.1:6653          127.0.0.1:45224         ESTABLISHED 1705/controller
tcp     4588      0 127.0.0.1:6653          127.0.0.1:45240         ESTABLISHED 1705/controller
tcp        0      0 127.0.0.1:45226         127.0.0.1:6653          ESTABLISHED 822/ovs-vswitchd
------------------------------------------------------------------------------------
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --if-exists del-br s1 -- --if-exists del-br s10 -- --if-exists del-br s11 -- --if-exists del-br s12 -- --if-exists del-br s13 -- --if-exists del-br s14 -- --if-exists de
l-br s15 -- --if-exists del-br s16 -- --if-exists del-br s17 -- --if-exists del-br s18 -- --if-exists del-br s19 -- --if-exists del-br s2 -- --if-exists del-br s20 -- --if-exists
del-br s3 -- --if-exists del-br s4 -- --if-exists del-br s5 -- --if-exists del-br s6 -- --if-exists del-br s7 -- --if-exists del-br s8 -- --if-exists del-br s9
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
( ip link del s1-eth1;ip link del s2-eth1;ip link del s3-eth1;ip link del s4-eth1;ip link del s5-eth1;ip link del s6-eth1;ip link del s7-eth1;ip link del s8-eth1;ip link del s9-et
h1;ip link del s10-eth1;ip link del s11-eth1;ip link del s12-eth1;ip link del s13-eth1;ip link del s14-eth1;ip link del s15-eth1;ip link del s16-eth1;ip link del s17-eth1;ip link
del s18-eth1;ip link del s19-eth1;ip link del s20-eth1;ip link del s1-eth2;ip link del s2-eth2;ip link del s2-eth2;ip link del s1-eth2;ip link del s2-eth3;ip link del s3-eth2;ip l
ink del s3-eth2;ip link del s2-eth3;ip link del s3-eth3;ip link del s4-eth2;ip link del s4-eth2;ip link del s3-eth3;ip link del s4-eth3;ip link del s5-eth2;ip link del s5-eth2;ip
link del s4-eth3;ip link del s5-eth3;ip link del s6-eth2;ip link del s6-eth2;ip link del s5-eth3;ip link del s6-eth3;ip link del s7-eth2;ip link del s7-eth2;ip link del s6-eth3;ip
 link del s7-eth3;ip link del s8-eth2;ip link del s8-eth2;ip link del s7-eth3;ip link del s8-eth3;ip link del s9-eth2;ip link del s9-eth2;ip link del s8-eth3;ip link del s9-eth3;i
p link del s10-eth2;ip link del s10-eth2;ip link del s9-eth3;ip link del s10-eth3;ip link del s11-eth2;ip link del s11-eth2;ip link del s10-eth3;ip link del s11-eth3;ip link del s
12-eth2;ip link del s12-eth2;ip link del s11-eth3;ip link del s12-eth3;ip link del s13-eth2;ip link del s13-eth2;ip link del s12-eth3;ip link del s13-eth3;ip link del s14-eth2;ip
link del s14-eth2;ip link del s13-eth3;ip link del s14-eth3;ip link del s15-eth2;ip link del s15-eth2;ip link del s14-eth3;ip link del s15-eth3;ip link del s16-eth2;ip link del s1
6-eth2;ip link del s15-eth3;ip link del s16-eth3;ip link del s17-eth2;ip link del s17-eth2;ip link del s16-eth3;ip link del s17-eth3;ip link del s18-eth2;ip link del s18-eth2;ip l
ink del s17-eth3;ip link del s18-eth3;ip link del s19-eth2;ip link del s19-eth2;ip link del s18-eth3;ip link del s19-eth3;ip link del s20-eth2;ip link del s20-eth2;ip link del s19
-eth3 ) 2> /dev/null
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
```

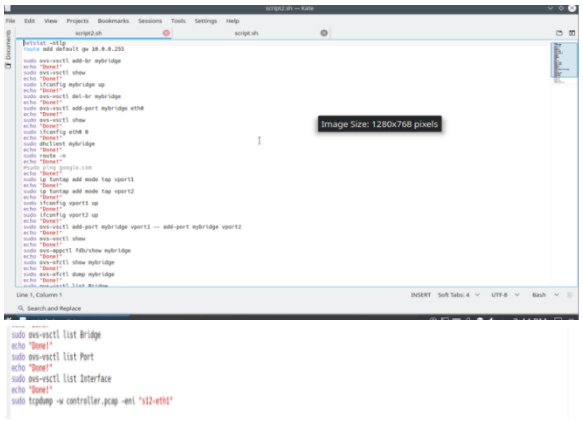
Fig. 5c. Output of open v-switch



```
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
ovs-ofctl: s2 is not a bridge or a socket
ovs-ofctl: s2 is not a bridge or a socket
ovs-vsctl: no row "s1" in table Bridge
------------------------------------------------------------
Server listening on UDP port 7575
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
```

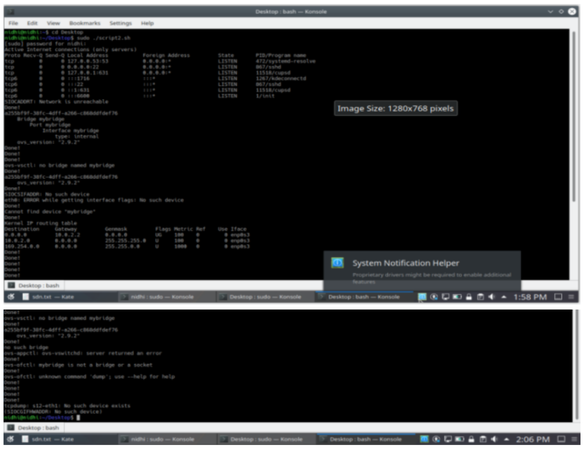
Fig. 5d. Output of open v-switch

Adding ovs-vsctl open v-switch libraries:



```bash
netstat -ntlp
route add default gw 10.0.0.255

sudo ovs-vsctl add-br mybridge
echo "Done!"
sudo ovs-vsctl show
echo "Done!"
sudo ifconfig mybridge up
echo "Done!"
sudo ovs-vsctl del-br mybridge
echo "Done!"
sudo ovs-vsctl add-port mybridge eth0
echo "Done!"
sudo ovs-vsctl show
echo "Done!"
sudo ifconfig eth0 0
echo "Done!"
sudo dhclient mybridge
echo "Done!"
sudo route -n
echo "Done!"
#sudo ping google.com
echo "Done!"
sudo ip tuntap add mode tap vport1
echo "Done!"
sudo ip tuntap add mode tap vport2
echo "Done!"
sudo ifconfig vport1 up
echo "Done!"
sudo ifconfig vport2 up
echo "Done!"
sudo ovs-vsctl add-port mybridge vport1 -- add-port mybridge vport2
echo "Done!"
sudo ovs-vsctl show
echo "Done!"
sudo ovs-appctl fdb/show mybridge
echo "Done!"
sudo ovs-ofctl show mybridge
echo "Done!"
sudo ovs-ofctl dump mybridge
echo "Done!"
sudo ovs-vsctl list Bridge
echo "Done!"
sudo ovs-vsctl list Port
echo "Done!"
sudo ovs-vsctl list Interface
echo "Done!"
sudo tcpdump -w controller.pcap -eni "s12-eth1"
```

Fig. 6a. install ovs-vsctl libraries install code

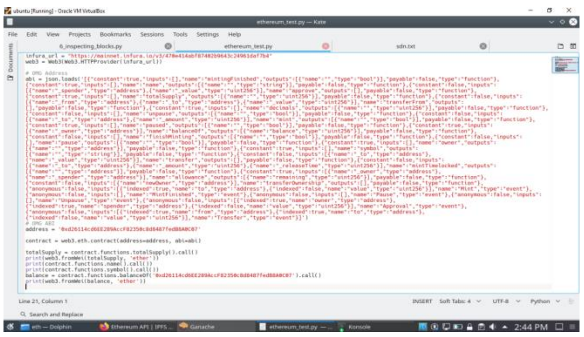


```
nidhi@nidhi:~$ cd Desktop
nidhi@nidhi:~/Desktop$ sudo ./script2.sh
[sudo] password for nidhi:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      472/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      867/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      11518/cupsd
tcp6       0      0 :::1716                 :::*                    LISTEN      1267/kdeconnectd
tcp6       0      0 :::22                   :::*                    LISTEN      867/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN      11518/cupsd
tcp6       0      0 :::6600                 :::*                    LISTEN      1/init
SIOCADDRT: Network is unreachable
Done!
a255bf9f-38fc-4dff-a266-c868ddfdef76
    Bridge mybridge
        Port mybridge
            Interface mybridge
                type: internal
    ovs_version: "2.9.2"
Done!
Done!
Done!
ovs-vsctl: no bridge named mybridge
Done!
a255bf9f-38fc-4dff-a266-c868ddfdef76
    ovs_version: "2.9.2"
Done!
SIOCSIFADDR: No such device
eth0: ERROR while getting interface flags: No such device
Done!
Cannot find device "mybridge"
Done!
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
Done!
Done!
Done!
Done!
Done!
```

```
Done!
ovs-vsctl: no bridge named mybridge
Done!
a255bf9f-38fc-4dff-a266-c868ddfdef76
    ovs_version: "2.9.2"
Done!
no such bridge
ovs-appctl: ovs-vswitchd: server returned an error
Done!
ovs-ofctl: mybridge is not a bridge or a socket
Done!
ovs-ofctl: unknown command 'dump'; use --help for help
Done!
Done!
Done!
Done!
tcpdump: s12-eth1: No such device exists
(SIOCGIFHWADDR: No such device)
nidhi@nidhi:~/Desktop$
```

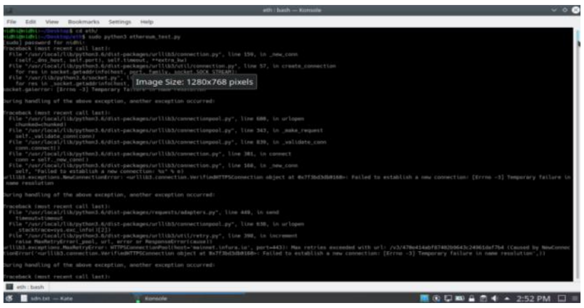
Fig. 6b. output of ovs-vsctl installed libraries

Ethereum_Test Code to test the ethereum:

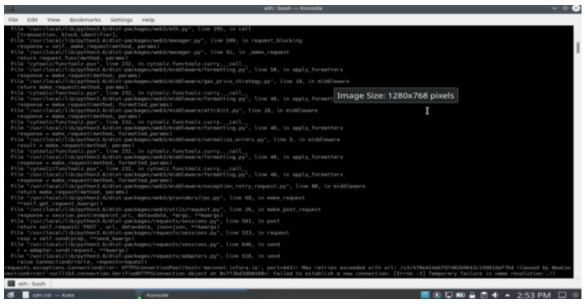


```python
infura_url = "https://mainnet.infura.io/v3/470e414abf87482b9643c24961daf7b4"
web3 = Web3(Web3.HTTPProvider(infura_url))

# OMG Address
abi = json.loads('[{"constant":true,"inputs":[],"name":"mintingFinished","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},
{"constant":true,"inputs":[],"name":"name","outputs":[{"name":"","type":"string"}],"payable":false,"type":"function"},{"constant":false,"inputs":
[{"name":"_spender","type":"address"},{"name":"_value","type":"uint256"}],"name":"approve","outputs":[],"payable":false,"type":"function"},
{"constant":true,"inputs":[],"name":"totalSupply","outputs":[{"name":"","type":"uint256"}],"payable":false,"type":"function"},{"constant":false,"inputs":
[{"name":"_from","type":"address"},{"name":"_to","type":"address"},{"name":"_value","type":"uint256"}],"name":"transferFrom","outputs":
[],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"decimals","outputs":[{"name":"","type":"uint256"}],"payable":false,"type":"function"},
{"constant":false,"inputs":[],"name":"unpause","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":false,"inputs":
[{"name":"_to","type":"address"},{"name":"_amount","type":"uint256"}],"name":"mint","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},
{"constant":true,"inputs":[],"name":"paused","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inputs":
[{"name":"_owner","type":"address"}],"name":"balanceOf","outputs":[{"name":"balance","type":"uint256"}],"payable":false,"type":"function"},
{"constant":false,"inputs":[],"name":"finishMinting","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":false,"inputs":
[],"name":"pause","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"owner","outputs":
[{"name":"","type":"address"}],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"symbol","outputs":
[{"name":"","type":"string"}],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"_to","type":"address"},
{"name":"_value","type":"uint256"}],"name":"transfer","outputs":[],"payable":false,"type":"function"},{"constant":false,"inputs":
[{"name":"_to","type":"address"},{"name":"_amount","type":"uint256"},{"name":"_releaseTime","type":"uint256"}],"name":"mintTimelocked","outputs":
[{"name":"","type":"address"}],"payable":false,"type":"function"},{"constant":true,"inputs":[{"name":"_owner","type":"address"},
{"name":"_spender","type":"address"}],"name":"allowance","outputs":[{"name":"remaining","type":"uint256"}],"payable":false,"type":"function"},
{"constant":false,"inputs":[{"name":"newOwner","type":"address"}],"name":"transferOwnership","outputs":[],"payable":false,"type":"function"},
{"anonymous":false,"inputs":[{"indexed":true,"name":"to","type":"address"},{"indexed":false,"name":"value","type":"uint256"}],"name":"Mint","type":"event"},
{"anonymous":false,"inputs":[],"name":"MintFinished","type":"event"},{"anonymous":false,"inputs":[],"name":"Pause","type":"event"},{"anonymous":false,"inputs":
[],"name":"Unpause","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"name":"owner","type":"address"},
{"indexed":true,"name":"spender","type":"address"},{"indexed":false,"name":"value","type":"uint256"}],"name":"Approval","type":"event"},
{"anonymous":false,"inputs":[{"indexed":true,"name":"from","type":"address"},{"indexed":true,"name":"to","type":"address"},
{"indexed":false,"name":"value","type":"uint256"}],"name":"Transfer","type":"event"}]')
# OMG ABI
address = '0xd26114cd6EE289AccF82350c8d8487fedB8A0C07'

contract = web3.eth.contract(address=address, abi=abi)

totalSupply = contract.functions.totalSupply().call()
print(web3.fromWei(totalSupply, 'ether'))
print(contract.functions.name().call())
print(contract.functions.symbol().call())
balance = contract.functions.balanceOf('0xd26114cd6EE289AccF82350c8d8487fedB8A0C07').call()
print(web3.fromWei(balance, 'ether'))
```

Fig. 7a. Ethereum_test code



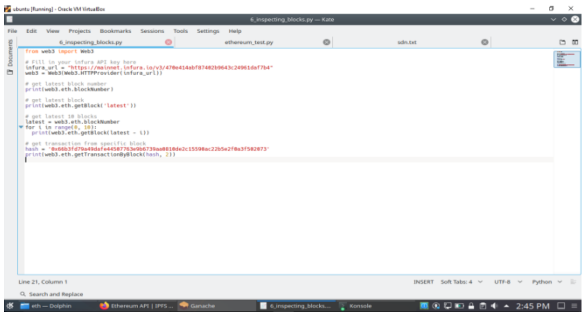
```
nidhi@nidhi:~/Desktop$ cd eth/
nidhi@nidhi:~/Desktop/eth$ sudo python3 ethereum_test.py
[sudo] password for nidhi:
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/urllib3/connection.py", line 159, in _new_conn
    (self._dns_host, self.port), self.timeout, **extra_kw)
  File "/usr/local/lib/python3.6/dist-packages/urllib3/util/connection.py", line 57, in create_connection
    for res in socket.getaddrinfo(host, port, family, socket.SOCK_STREAM):
  File "/usr/lib/python3.6/socket.py", l
    for res in _socket.getaddrinfo(host,
socket.gaierror: [Errno -3] Temporary fa

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/urllib3/connectionpool.py", line 600, in urlopen
    chunked=chunked)
  File "/usr/local/lib/python3.6/dist-packages/urllib3/connectionpool.py", line 343, in _make_request
    self._validate_conn(conn)
  File "/usr/local/lib/python3.6/dist-packages/urllib3/connectionpool.py", line 839, in _validate_conn
    conn.connect()
  File "/usr/local/lib/python3.6/dist-packages/urllib3/connection.py", line 301, in connect
    conn = self._new_conn()
  File "/usr/local/lib/python3.6/dist-packages/urllib3/connection.py", line 168, in _new_conn
    self, "Failed to establish a new connection: %s" % e)
urllib3.exceptions.NewConnectionError: <urllib3.connection.VerifiedHTTPSConnection object at 0x7f3bd3db0160>: Failed to establish a new connection: [Errno -3] Temporary failure in
 name resolution

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/requests/adapters.py", line 449, in send
    timeout=timeout
  File "/usr/local/lib/python3.6/dist-packages/urllib3/connectionpool.py", line 638, in urlopen
    _stacktrace=sys.exc_info()[2])
  File "/usr/local/lib/python3.6/dist-packages/urllib3/util/retry.py", line 398, in increment
    raise MaxRetryError(_pool, url, error or ResponseError(cause))
urllib3.exceptions.MaxRetryError: HTTPSConnectionPool(host='mainnet.infura.io', port=443): Max retries exceeded with url: /v3/470e414abf87482b9643c24961daf7b4 (Caused by NewConnec
tionError('<urllib3.connection.VerifiedHTTPSConnection object at 0x7f3bd3db0160>: Failed to establish a new connection: [Errno -3] Temporary failure in name resolution',))

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
```



```
[illegible]
```

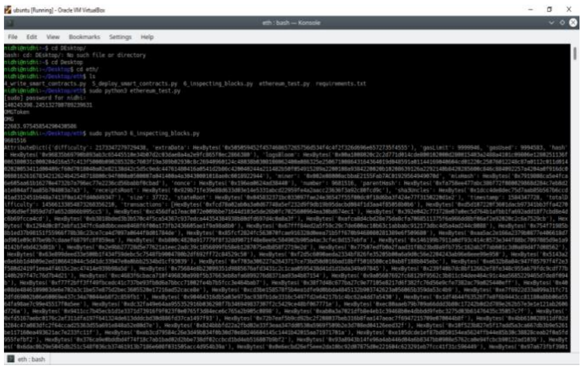
Fig. 7b. Ethereum_test code output

Inspecting blocks over ethereum:



```
from web3 import Web3

# Fill in your infura API key here
infura_url = "https://mainnet.infura.io/v3/470e414abf87402b9643c24961daf7b4"
web3 = Web3(Web3.HTTPProvider(infura_url))

# get latest block number
print(web3.eth.blockNumber)

# get latest block
print(web3.eth.getBlock('latest'))

# get latest 10 blocks
latest = web3.eth.blockNumber
for i in range(0, 10):
  print(web3.eth.getBlock(latest - i))

# get transaction from specific block
hash = '0x66b3fd79a49dafe44507763e9b6739aa0010de2c15590ac22b5e2f0a3f502073'
print(web3.eth.getTransactionByBlock(hash, 2))
```

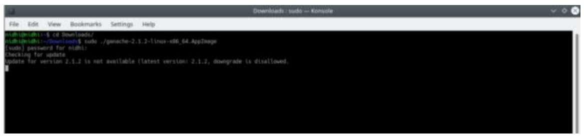
Fig. 8a. Inspecting_blocks code



```
nidhi@nidhi:~$ cd DEsktop/
bash: cd: DEsktop/: No such file or directory
nidhi@nidhi:~$ cd Desktop
nidhi@nidhi:~/Desktop$ cd eth/
nidhi@nidhi:~/Desktop/eth$ ls
4_write_smart_contracts.py  5_deploy_smart_contracts.py  6_inspecting_blocks.py  ethereum_test.py  requirements.txt
nidhi@nidhi:~/Desktop/eth$ sudo python3 ethereum_test.py
[sudo] password for nidhi:
140245398.24513270789239631
OMGToken
OMG
22683.975458542904305886
nidhi@nidhi:~/Desktop/eth$ sudo python3 6_inspecting_blocks.py
9681516
AttributeDict({'difficulty': 2173347279729430, 'extraData': HexBytes('0x505059452f457460657265756d534f4c4f2f326d696e6572735f4555 [illegible]
[illegible]
```

Fig. 8b. Inspecting_blocks code output

Ganache needs to be downloaded and installed. It is locally deployed in ethereum. Ganache screen will be displayed where outputs regarding transactions, records, accounts etc.are displayed.



```
nidhi@nidhi:~$ cd Downloads/
nidhi@nidhi:~/Downloads$ sudo ./ganache-2.1.2-linux-x86_64.AppImage
[sudo] password for nidhi:
Checking for update
Update for version 2.1.2 is not available (latest version: 2.1.2, downgrade is disallowed.
```

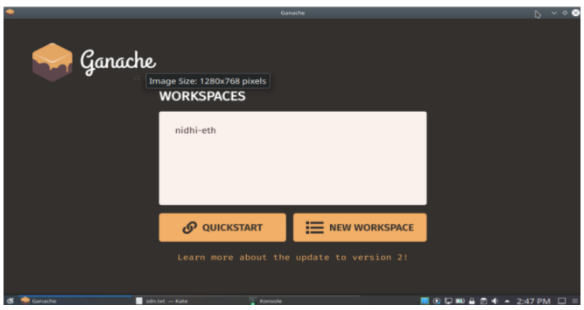
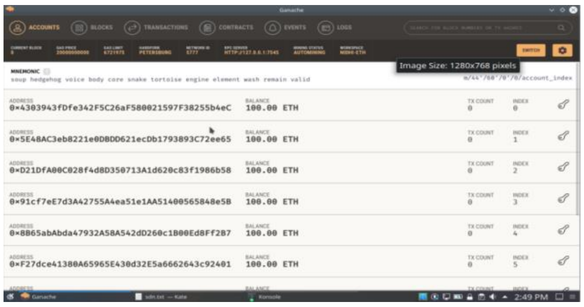


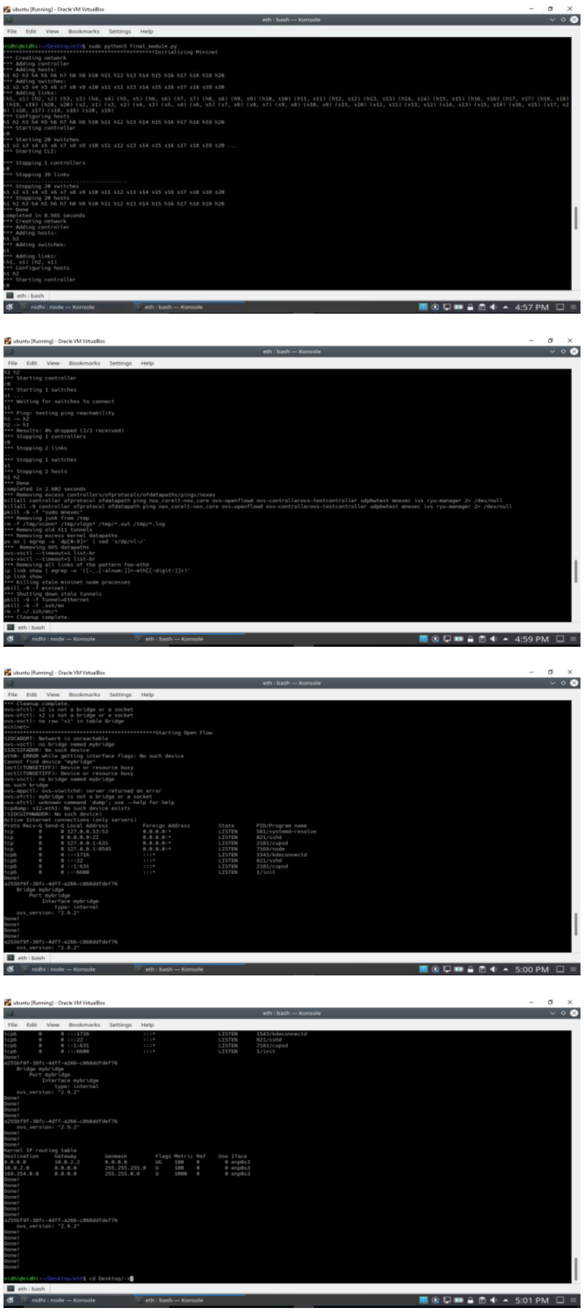
Fig. 9a. code to access ganache



```
MNEMONIC
soup hedgehog voice body core snake tortoise engine element wash remain valid        m/44'/60'/0'/0/account_index

ADDRESS                                          BALANCE          TX COUNT   INDEX
0×4303943fDfe342F5C26aF580021597F38255b4eC       100.00 ETH       0          0
ADDRESS                                          BALANCE          TX COUNT   INDEX
0×5E48AC3eb8221e0DBDD621ecDb1793893C72ee65       100.00 ETH       0          1
ADDRESS                                          BALANCE          TX COUNT   INDEX
0×D21DfA00C028f4d8D350713A1d620c83f1986b58       100.00 ETH       0          2
ADDRESS                                          BALANCE          TX COUNT   INDEX
0×91cf7eE7d3A42755A4ea51e1AA51400565848e5B       100.00 ETH       0          3
ADDRESS                                          BALANCE          TX COUNT   INDEX
0×8B65abAbda47932A58A542dD260c1B00Ed8Ff2B7       100.00 ETH       0          4
ADDRESS                                          BALANCE          TX COUNT   INDEX
0×F27dce41380A65965E430d32E5a6662643c92401       100.00 ETH       0          5
```

Fig. 9b. outputs regarding transactions and records of various accounts

Final module output:



```
nidhi@nidhi:~/Desktop/eth$ sudo python3 final_module.py
************************************************Initializing Mininet
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (h7, s7) (h8, s8) (h9, s9) (h10, s10) (h11, s11) (h12, s12) (h13, s13) (h14, s14) (h15, s15) (h16, s16) (h17, s17) (h18, s18)
 (h19, s19) (h20, s20) (s2, s1) (s3, s2) (s4, s3) (s5, s4) (s6, s5) (s7, s6) (s8, s7) (s9, s8) (s10, s9) (s11, s10) (s12, s11) (s13, s12) (s14, s13) (s15, s14) (s16, s15) (s17, s1
6) (s18, s17) (s19, s18) (s20, s19)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
*** Starting controller
c0
*** Starting 20 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 ...
*** Starting CLI:

*** Stopping 1 controllers
c0
*** Stopping 39 links
.......................................
*** Stopping 20 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20
*** Stopping 20 hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
*** Done
completed in 8.565 seconds
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
```

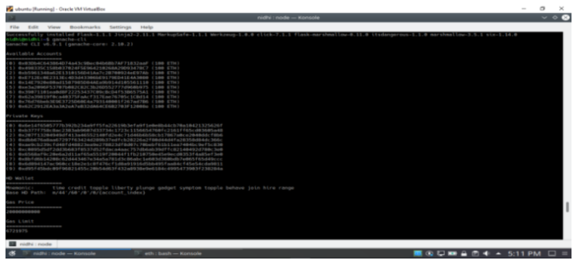
```
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 2.602 seconds
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_corelt-nox_core ovs-openflowd ovs-controllerovs-testcontroller udpbwtest mnexec ivs ryu-manager 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```

```
*** Cleanup complete.
ovs-ofctl: s2 is not a bridge or a socket
ovs-ofctl: s2 is not a bridge or a socket
ovs-vsctl: no row "s1" in table Bridge
mininet>
************************************************Starting Open Flow
SIOCADDRT: Network is unreachable
ovs-vsctl: no bridge named mybridge
SIOCSIFADDR: No such device
eth0: ERROR while getting interface flags: No such device
Cannot find device "mybridge"
ioctl(TUNSETIFF): Device or resource busy
ioctl(TUNSETIFF): Device or resource busy
ovs-vsctl: no bridge named mybridge
no such bridge
ovs-appctl: ovs-vswitchd: server returned an error
ovs-ofctl: mybridge is not a bridge or a socket
ovs-ofctl: unknown command 'dump'; use --help for help
tcpdump: s12-eth1: No such device exists
(SIOCGIFHWADDR: No such device)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      501/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      821/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      2181/cupsd
tcp        0      0 127.0.0.1:8545          0.0.0.0:*               LISTEN      7169/node
tcp6       0      0 :::1716                 :::*                    LISTEN      1543/kdeconnectd
tcp6       0      0 :::22                   :::*                    LISTEN      821/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN      2181/cupsd
tcp6       0      0 :::6600                 :::*                    LISTEN      1/init
Done!
a255bf9f-38fc-4dff-a266-c868ddfdef76
    Bridge mybridge
        Port mybridge
            Interface mybridge
                type: internal
    ovs_version: "2.9.2"
Done!
Done!
Done!
Done!
a255bf9f-38fc-4dff-a266-c868ddfdef76
    ovs_version: "2.9.2"
```

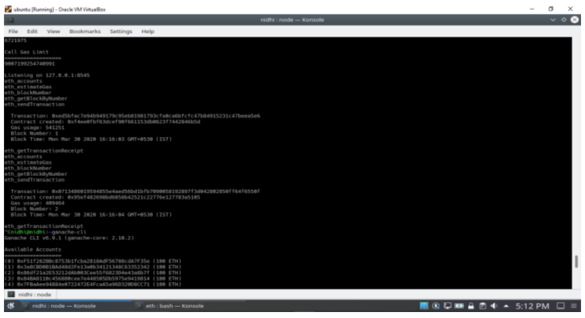
```
tcp6       0      0 :::1716                 :::*                    LISTEN      1543/kdeconnectd
tcp6       0      0 :::22                   :::*                    LISTEN      821/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN      2181/cupsd
tcp6       0      0 :::6600                 :::*                    LISTEN      1/init
Done!
a255bf9f-38fc-4dff-a266-c868ddfdef76
    Bridge mybridge
        Port mybridge
            Interface mybridge
                type: internal
    ovs_version: "2.9.2"
Done!
Done!
Done!
Done!
a255bf9f-38fc-4dff-a266-c868ddfdef76
    ovs_version: "2.9.2"
Done!
Done!
Done!
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
Done!
Done!
Done!
Done!
Done!
Done!
Done!
a255bf9f-38fc-4dff-a266-c868ddfdef76
    ovs_version: "2.9.2"
Done!
Done!
Done!
Done!
Done!
Done!
Done!

nidhi@nidhi:~/Desktop/eth$ cd Desktop/~x
```

Fig. 10. Final module output

Final ganache output:



```
Successfully installed Flask-1.1.1 Jinja2-2.11.1 MarkupSafe-1.1.1 Werkzeug-1.0.0 click-7.1.1 flask-marshmallow-0.11.0 itsdangerous-1.1.0 marshmallow-3.5.1 six-1.14.0
nidhi@nidhi:~$ ganache-cli
Ganache CLI v6.9.1 (ganache-core: 2.10.2)

Available Accounts
==================
(0) 0x03Db4C643864D74a43c9Bec04b6Bb7AF71832aaF (100 ETH)
(1) 0x498335C158b037024F5E964210268A29D93478C7 (100 ETH)
(2) 0xb5961348a62E1310156D41Aa7c28780924eE97Ab (100 ETH)
(3) 0xE712Ec0E2313Ec4D3d433066E9179ED41E4A3000 (100 ETH)
(4) 0x14E7920e80ad15079850B4AEa90914d105561110 (100 ETH)
(5) 0xe3a2096F53707b082C82C3b26DD552777d960b975 (100 ETH)
(6) 0x39071101ea0d8F22253437C09cBcD4f53B6575A1 (100 ETH)
(7) 0x62a39819f0ca40375FaAcf317Eae76705c1C0d14 (100 ETH)
(8) 0x76d76beb3E9E3725D60E4a793140001f267ad7B6 (100 ETH)
(9) 0x62C2912EA3a3A2eA7e832dA64CE6B2703F12008e (100 ETH)

Private Keys
==================
(0) 0x6e14f6505777b392b234a9ff5fa22619b3efa9f1e0e0b44cb70a104213256 26f
(1) 0xb377f758c8ac2383ab9607d33734c1723c1156654760fc2161ff65cd03605a48
(2) 0x207f13204949df413a46552140fd2e4c71d46b6b58cb17067a0ce2040ddcf0b6
(3) 0xdbb676a0aa67297f63424d200b37edfcb20226a2f80d44d4fa20358d04dc366c
(4) 0xae9cb239cfd40fd40823ea9e270823df0d07c70bebf61b11ea7404bc9ef5c830
(5) 0xc0095d5df2dd3b603f0537d52fdaca4aac757db6ab39dffc02140492d700c3e0
(6) 0x6568af9c20e6a2d11af65a5519f20044f1fb210750e45e9ecd0353f4a85ef3e0
(7) 0x8bfd6b14208c62d443467e34a5a781d3c06abc1e603d360bdb7e065f65d49ccc
(8) 0x6d094147ac960cc18e2e1c0f476cf1d8a91916d5bb495faa84cf45e54cda9011
(9) 0xd95f45bdc09f96021455c20b54d63f432a0930e9e6184c4995473903f238204a

HD Wallet
==================
Mnemonic:      time credit topple liberty plunge gadget symptom topple behave join hire range
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
==================
20000000000

Gas Limit
==================
6721975
```



```
6721975

Call Gas Limit
==================
9007199254740991

Listening on 127.0.0.1:8545
eth_accounts
eth_estimateGas
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0xed5bfac7e94b949179c95eb81981793cfa0ca6bfcfc47b84915231c47beea5e6
  Contract created: 0xf4ee0fbf63dcef90f661153db0623f7442846b5d
  Gas usage: 541251
  Block Number: 1
  Block Time: Mon Mar 30 2020 16:16:03 GMT+0530 (IST)

eth_getTransactionReceipt
eth_accounts
eth_estimateGas
eth_blockNumber
eth_getBlockByNumber
eth_sendTransaction

  Transaction: 0x0713486019594855e4aed56bd1bfb7090058192897f3d042882850ff64f6550f
  Contract created: 0x95ef482698bd6858b42521c22776e127783a5105
  Gas usage: 409464
  Block Number: 2
  Block Time: Mon Mar 30 2020 16:16:04 GMT+0530 (IST)

eth_getTransactionReceipt
^Cnidhi@nidhi:~ganache-cli
Ganache CLI v6.9.1 (ganache-core: 2.10.2)

Available Accounts
==================
(0) 0xF51f282B8c8753b1fcba2818AdF56788cdA7F35e (100 ETH)
(1) 0x3a0CBD0B1BAd48d2Fe13a0b34121340C63352342 (100 ETH)
(2) 0x86df21a2E53212dAb003Cee55f6823D4e43a8b7f (100 ETH)
(3) 0x84BA8110c456880cee7e4485050b5975e9419814 (100 ETH)
(4) 0x7FBaAee94884e0722472E4FcaA5a96D320D8CC71 (100 ETH)
```

Fig. 11. Final ganache output.

V. ADVANTAGE

A. Background

1.Majority of SDN improvement works on characteristics , not security. Those SDN are endangered from new assault vectors that were literally not viable initially compared to traditional network[6][7].

2.Hosts/servers of network might generally be exposed to assault in traditional network,. But at present with SDN, new APIs & hence susceptibilities occur for network alone.

3.To unveil this issue, earstwhile a sole rogue element alongwith a transfer or compute component, infused with help of hacker is usual. With help of a SDN, the hacker may be able to see, replicate, manipulate, distort communications on the network.

4. Hence any security solution has to be capable of scaling and function the performance to permit various valid elements in altogether while refusing a sole rogue detail from hacker.

B. Solution

A solution wherein some factor that takes vicinity on SDN is trapped in legally verifiable & invariable log–the blockchain. In case cyberpunk attempt & cover his trail byway of further hacking inside the log server & altering the record of actions, owing to the fact the blockchain & its data prevail in many locations without deferment so that each modification would be denied before blockchain peers.

VI. LITERATURE SURVEY

In modern day, various studies are implemented in SDN for determining its potential to improve cease-to-stop network security. Various strategies are recommended to accomplish various security acts for utilizing firewall proposition because it miles the foremost aspect to safeguard spiteful assaults at the network. SDN controllers such as RYU, floodlight, POX, etc. have extended aid to firewall building blocks for try out & evolution. fundamental design of firewall in SDN surroundings is shown in Fig. 12.

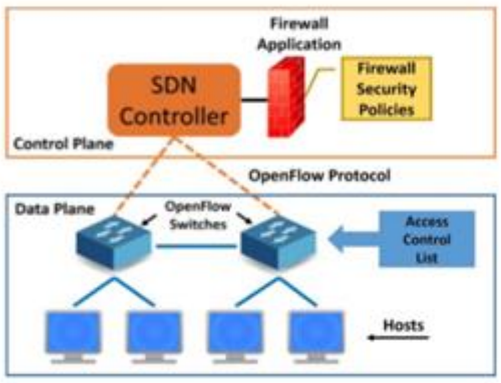


Fig. 12. The fundamental design of firewall in SDN surroundings.

A concept of reactive stateful firewall & techniques for improving it's functioning with the use of S-Table & SecPolTable alongwith the flow table was suggested by Nife et al. in a latest study[9][10]. But, obscurities about the execution and validating the recommended idea still exist. In a further research, a reactive stateful firewall (using raspberry pi) which expands throughout by diffusing network flow across various links comprising firewalls (for instigating sceptical traffic) & an additional link (escaping the trustful traffic) was developed by Vasaka et al[11][12].

A fresh proposal for dynamic stateful firewall in SDN surroundings with an instigator as a network protection control system operating on application plane was initiated by Zerkane et al[9]. Instigator is able to control different controllers & implement protection measures in firewall operation existing in controllers. Tran and Ahn introduced a flowtracker stateful firewall having capability of learning network topography & implement protection measures so as to lower useless admittance in flow table which filtrate network traffic[13].

VII. PROPOSED FRAMEWORK

The SDN framework comprises of network controllers strolling at manage plane, various data forwarding devices (e.g. Switches, routers, etc.) at information plane and network programs running on top of controllers. The network applications instruct controllers to implement community facilities that cover switching, routing, load balancing, firewall, etc. OpenFlow protocol prevail between manage plane & facts plane to establish connection among controller and statistics plane devices. The model 1.0 of openflow had basic 12 in shape fields and one go with the flow desk. The recommended framework is depends totally on the latest variant of the OpenFlow (model 1.5) that serves 44 strong fields and a pair of float tables. Although open flow ver 1.5 was launched around 2014 [9], research on this subect is constricted. The suggested framework is the application of firewall viability on an open-source SDN controller working on openflow ver 1.5 guidelines. Four openflow authorized switches linking 4 different person systems are utilized so as to confirm scalability. The Ryu framework is working as an SDN controller as it assists openflow model 1.5 specs. The firewall directives are specified to both allow or obstruct the packets centred completely at the header data together with origin & destination mac cope with, IPv4/IPv6 deal with, port numbers & so on. Since those particulars will control destiny openflow authorised devices, such firewall directives are typically built on the suit fields provided in openflow switch particulars ver 1.5. The community controller can lay directives onto each character switch within the firewall utility because the firewall module fetch data about the associated and accessible switches within the community. The firewall software continuously video display units, get liberty of entry to control listing set up on switches to ensure it isn't often altered by any outer or internal device and after revelation, utility re-route the flows in network as precautionary measure.

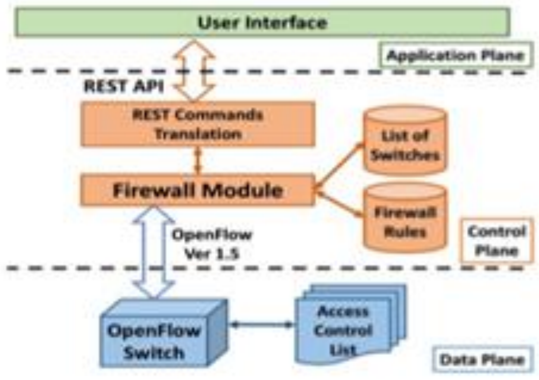


Fig. 13. The block diagram of SDN firewall applications

A. Building Blocks of Firewall Model

The fig. 2 depicts building blocks of SDN firewall using firewall application working on control plane in SDN controller when it is getting linked to openflow switches. Firewall Module, REST commands translation, record of switches and firewall principle are four main elements of the firewall application. The firewall module is centre of the firewall application which co-ordinates with controller module for executing firewall principle in the network tools. Firewall policies on user interface can be approached, set, remove or alter by the network administrator through REST Application Program Interface.

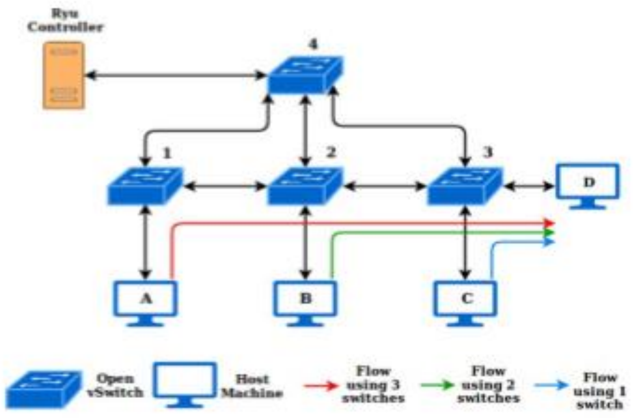


Fig. 3. (1) Traffic flow through 3 switches, (2) Traffic flow through 2 switches, (3) Traffic flow through 1 switch

Fig. 14. Openflow Switches

The protection scheme of the network is governed by a list of firewall directives and it comprises of independent directives for respective switches or separate VLANs in a switch. On data plane, each switch comprises of access control list and is filled with various firewall directives acquired from firewall application. Those firewall directives depends on the match fields prescribed by OpenFlow Switch Specifications ver 1.5.

VIII. CONCLUSION

SDN has modernized flexile network monitoring whilst presenting programmability for superior command on data plane configuration. In addition to integrate MAC/IP/TCP layer element in plain data forwarding gadget, openflow protocol has allowed precise packet filtering. This assists in the application of wide network protection schemes without influencing the network functioning in a vast network. SDN based firewall could be significant technic for protecting malicious risks in vast networks is shown by the tests performed on the prototype network for three distinctive packet types i.e. ICMP, TCP and UDP. The intended firewall is justified on GNS3 platform, and executing such firewall with OpenFlow v1.5 has uplifted the aspirations to include upcoming versions of OpenFlow protocol for superior security visions. Apart from executing firewall protection schemes, the application may comprise protection qualities such as deep packet scrutiny, incursion identification to strengthen safety of the network.

In this research, we advocate a centralized security framework based on blockchain above cloud environment in SDN-enabled ethereum blockchain. Utilising the rigid characteristics of blockchain, the liability of the source message is verified. With the help of framework based on blockchain, we offer trust management for the vehicular system in case where malicious nodes can claim fake messages or messages can be tempered. Both theoretic studies and experimentation outcome explain the effectiveness of our framework since the diagnosis precision of the malicious nodes are notably improved.

REFERENCES

[1]. Q. Jing ,Athanasios, Vasilakos, J. Wan, “Security of the Internet of Things: perspectives and challenges”. In: Wireless Networks, vol. 20, pp 2481–2501, Springer link(2014)

[2]. R. Biswas, R. Giaffreda, “IoT and Cloud Convergence: Opportunities and Challenges”. In: IEEE World Forum on Internet of Things (WF-IoT) (2014)

[3]. F. Olivier, G. Carlos, N. Florent, “New Security Architecture for IoT Network”, In: International Workshop on Big Data and Data Mining Challenges on IoT and Pervasive Systems (BigD2M 2015).

[4]. D. Levin et al., “Logically Centralized?: State Distribution Trade-offs in Software Defined Networks,” Proc. 1st ACM SIGCOMM Wksp . Hot Topics in Software Defined Networks, Aug. 2012, pp. 1–6

[5]. X. Wu et al., “A Multipath Resource Updating Approach for Distributed Controllers in the Software-Defined Network.,” Science China Info. Sciences, vol. 59, no. 9, Sept. 2016, pp. 92,301–10

[6]. Z. Qingyun et al., “On Generally of the Data Plane and Scalability of the Control Plane in Software-Defined Networking,” China Commun., vol. 11, no. 2, Feb. 2014, pp. 55–64

[7]. K. Christidis and M. Devetsikiotis, “Blockchains and SmartContracts for the Internet of Things,” IEEE Access, vol. 4,May 2016, pp. 2292–303.

[8]. M. Tsagkaropoulos, I. Politis, C. Tselios, T. Dagiuklas, and S. Kotsopoulos, “Service continuity over intertechnology rats,” in 2011 IEEE 16th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), June 2011, pp. 117–121.

[9].] J. M. Batalla et al., “A Novel Methodology for Efficient Throughput Evaluation in Virtualized Routers,” 2015 IEEE ICC, June 2015, pp. 6899–905.

[10]. Abdelaziz, A. T. Fong, A. Gani, U. Garba, S. Khan, A. Akhunzada,H. Talebian, and K.-K. R. Choo, “Distributed controller clustering in software defined networks,” PloS one, vol. 12, no. 4, 2017.

[11]. D. Chourishi, A. Miri, M. Milic, and S. Ismaeel, “Role-based multiple ´controllers for load balancing and security in sdn,” in 2015 IEEE Canada International Humanitarian Technology Conference (IHTC2015). IEEE, 2015, pp. 1–4.

[12]. S. W. Pritchard, G. P. Hancke, and A. M. Abu-Mahfouz, “Security in Software-Defined Wireless Sensor Networks: Threats, Challenges and Potential Solutions,” in 2017 IEEE 15th International Conference on Industrial Informatics (INDIN). IEEE, 2017, pp. 168–173.

[13]. P. K. Sharma, S. Singh, Y.-S. Jeong, and J. H. Park, “Distblocknet: A Distributed Blockchains-based Secure SDN Architecture for IoT Networks,” IEEE Communications Magazine, vol. 55, no. 9, pp. 78–85, 2017.