A

REPORT

on

“Securing Network Using Raspberry Pi by Implementing VPN, Pi-Hole, and IPS (VPiSec)”

भारतीय प्रौद्योगिकी संस्थान हैदराबाद

Indian Institute of Technology Hyderabad

Course

(CS6220) Topics in Network

Date

23/04/2022

Submitted by

Group No. 4

Nilesh Kale (CS21MTECH11022)
Devang Dubey (CS21 MTECH14013)
Amit Kumar (CS21MTECH11007)

Index

I. INTRODUCTION

This paper presents an innovation of implementing a VPN router on a Raspberry Pi that is capable of encrypting the end-to-end connection between a user's device to servers so that the attacker cannot intercept the data. Also, a Pi-Hole is integrated to block advertisements at the user's device. Pi-Hole works as a DNS sinkhole that blocks the queries containing advertisements from being forwarded to the internet, unlike ad-block which can only hide the advertisements and only work with supported browsers. To add better security, OSSEC (open-source host-based intrusion detection system) IPS is also implemented on Raspberry-pi through which users are connected to VPN and Pi-hole. IPS detects and prevents known attacks like DoS or DDoS.

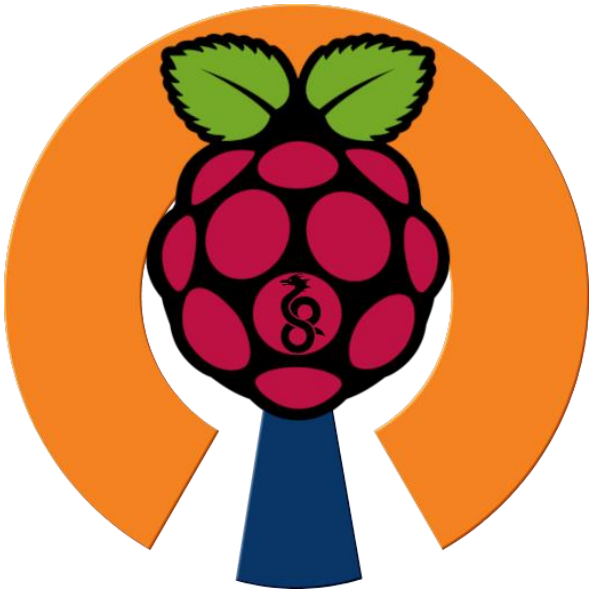
Problem statement

Barely users are sensitive that their data is being monitored by Internet Service Providers (ISPs) and other third-party companies. Every webpage they visit, each information they fill in the search box are being monitored by a third-party company that wants to know the user's interest and then will pop up advertisements which are related to the user's interest. This turns into the issue when the third party has all the sensitive information and misuses it in unethical manners. Thus, it is the reason why a device that can protect the user is needed. A device with a capability to conceal the user's Internet Protocol (IP) and protect the user from any tracker and advertisement from the internet is proposed in this paper.

II. BACKGROUND

RASPBERRY PI

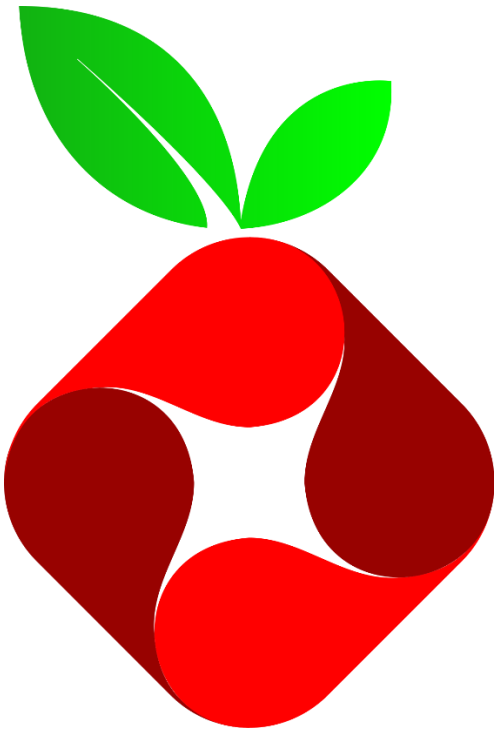
It is a well known small single based board computer suitable for IoT appplications. It can be used as a VPN server to a home network. It can also act like an advertisement blocker which prevents advertisements and popups from entering a user network. The project also uses it for the implementation.



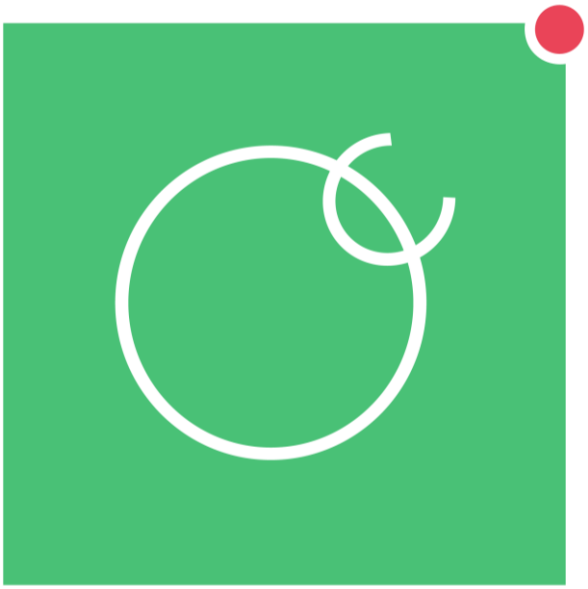
OpenVPN

OpenVPN is an open-source software application that implements VPN techniques between devices that are connected to the Internet. OpenVPN can transverse through firewalls and Network Address Translation (NAT). Moreover, OpenVPN is a cross-platform software that allows people to install and operate the software in various kinds of Operating System (OS). This is unlike any other VPN provider that always makes a VPN only for windows or Linux only. OpenVPN supports both Windows and Linux which makes it flexible and fully customizable.

Pi-Hole



Pi-hole blocks ads across a whole network, including most applications. Pi-hole is intended to use on embedded devices with network capability such as Raspberry Pi. There are different approach used by Adblock and Pi-hole to block ads. Adblock keep the web browser from viewing the ads from sites inside the browser. The ads are still being downloaded and still there but it is just being hidden by the Adblock. Pi-hole, on the other hand, does not even allow the Domain Name System (DNS) request for the ads to leave the network, which means that the ads do not even get downloaded to the user's network.



IDS-OpenCanary

It is a daemon that runs canary services, which trigger alerts when (ab)used. The alerts can be sent to a variety of sources, including syslog, emails and a companion daemon opencanary-correlator. The Correlator coalesces multiple related events into a single alert sent via email or SMS.

III. METHODOLOGY

DESIGN

The design phase includes the use of a graphical scenario for the situation of the problem and the solution for the scenario. After that, an illustration diagram was used to illustrate the design of the system architecture.

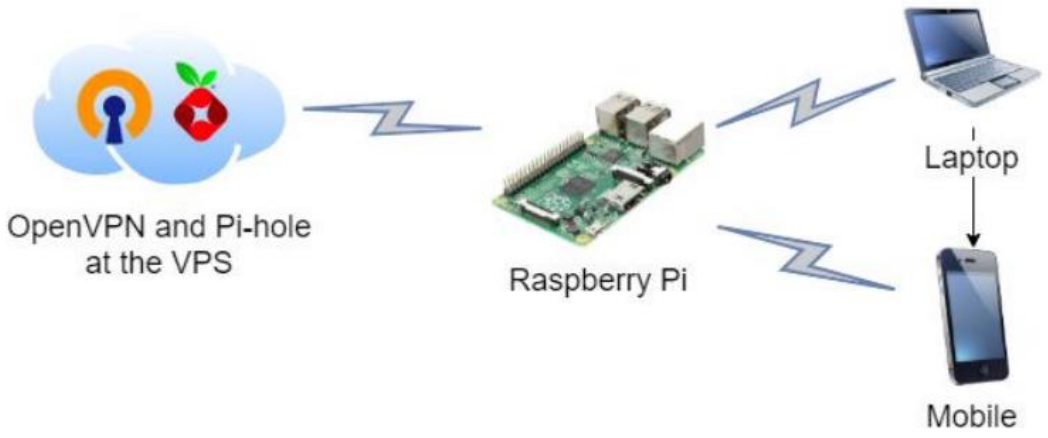


Figure shows the graphical scenario of securing network connection using Raspberry Pi. This system required the Raspberry Pi to connect to the VPN that has been configured on the Virtual Private Server (VPS). Pi-hole was installed alongside the VPN. The user devices needed to be configured so that the default gateway for the devices can be sent to the Raspberry Pi. The system protected all data which travels through the VPN, as the Raspberry Pi blocked any suspicious packets using IPS. All the traffic that travelled into the user will be filtered to check for any known attacks or annoying online advertisements. The suspicious packets used IPS. All the traffic that travels into the user will be filtered to check for any known attacks or an annoying online advertisement. The user will feel safe and can safely surf the internet without having to worry about any data leakage or any type of attack.

INSTALLATION & SETUP

The Raspberry Pi 3 B was equipped with Raspbian as its Operating System. After the installation for the OS is finished, remote connection features were set up to ease user jobs to maintain the Raspberry Pi and to create the bridge connection between public Wi-Fi access and user's device.

Pi-Hole

Method 1 (Automated Installer)

```
curl -sSL https://install.pi-hole.net | bash
```

Method 2 (Download and Execute)

```
wget -O basic-install.sh https://install.pi-hole.net
sudo bash basic-install.sh
```

Any settings we configure during installation can be updated later. At some point, it asks us to select an upstream DNS provider (Fig. 2). This is the server on which lookups of non-blocked hostnames will be performed.

Fig. 2: Select upstream DNS

Then, it will ask to select an adlist. leave the default on. Later, we can add more lists, including custom ones.

When the installation is finished, we get a summary message that includes the IP addresses of the Pi-hole and the randomly generated admin password (Fig. 9).

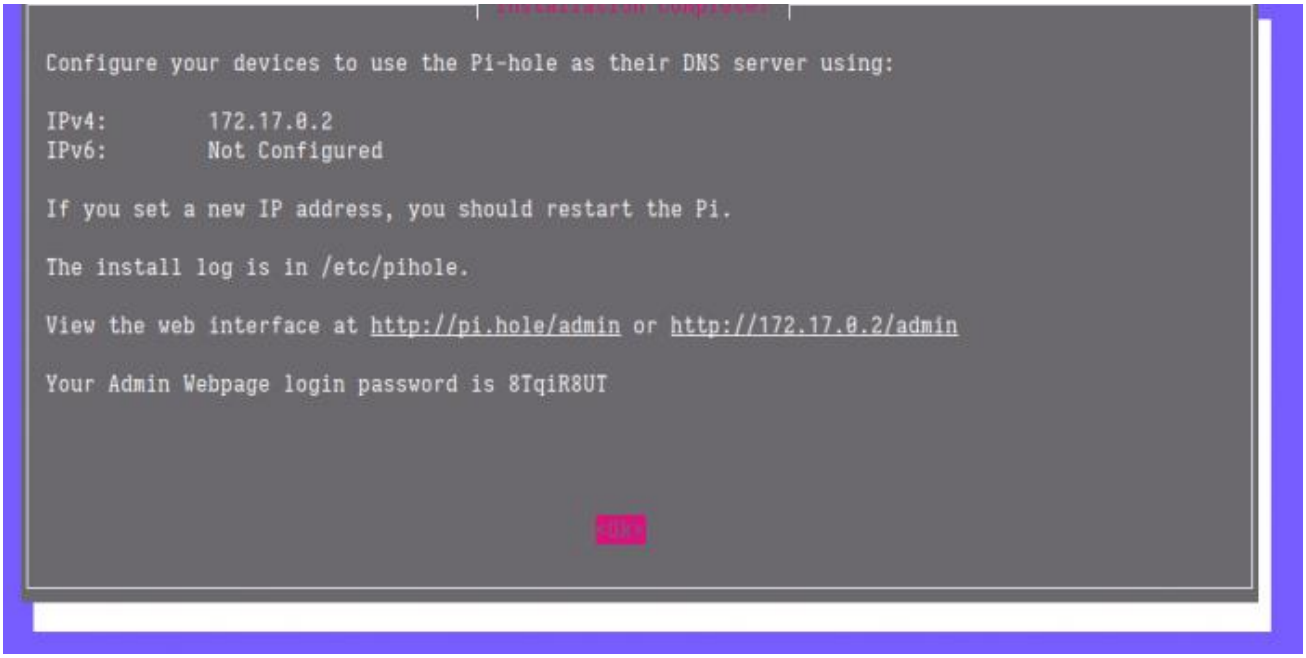


Fig. 9: Pi-hole installation summary

Click OK to conclude the installation. To be sure the installation succeeded, open a web browser and go to http://IP_ADDRESS/admin, where IP_ADDRESS is the IPv4 address of the Pi-hole device (Fig. 9). Note that the http://pihole/admin only works after we setup device to use the Pi-hole DNS server. Click on log-in and enter (randomly-generated) password. Now we can be in the Pi-hole admin panel (Fig. 10).

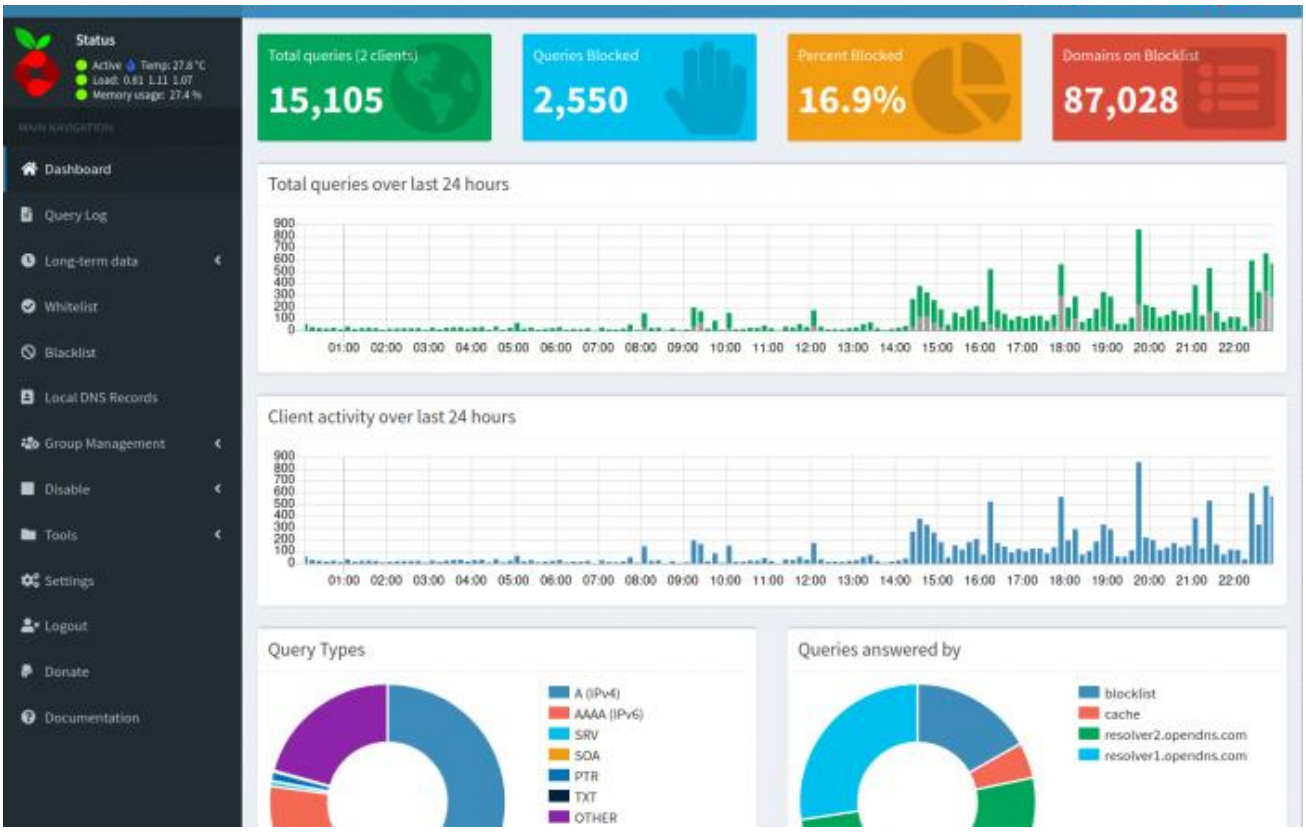


Fig. 10: Pi-hole admin panel

Now that we have Pi-hole installed, the last step is configuring network to use Pi-hole as its DNS server

The preferred method for doing this is to change our router's DNS server and point it to the Pi-hole IP address, ensuring any client that connects to the network receives the Pi-hole as its DNS server. Typically this requires us to access the router's administration panel. There, we should have a field to set the primary and secondary DNS servers. Set the primary address to the Pi-hole's IP address, and reset any open network connection we may have on the devices. Now, when we connect to home network, we should get the Pi-hole as the DNS server.

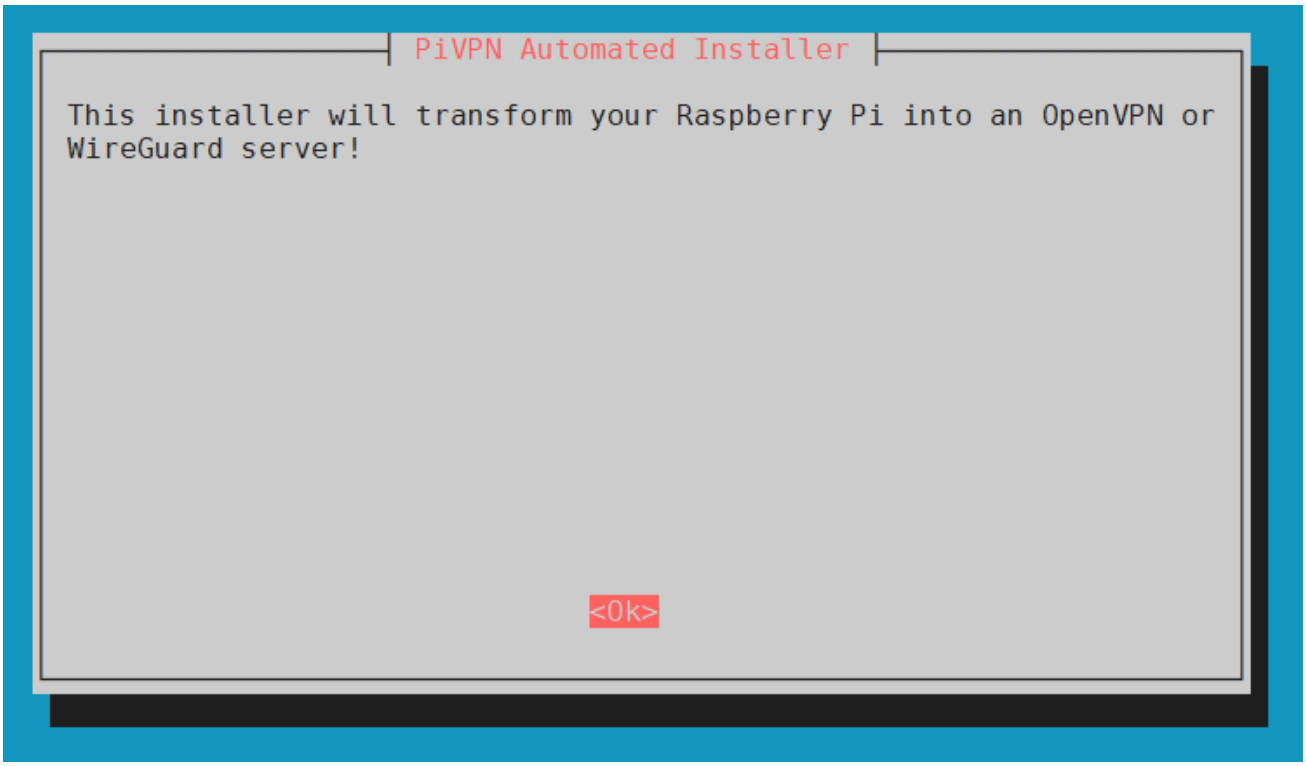
Pi-VPN

Step 1

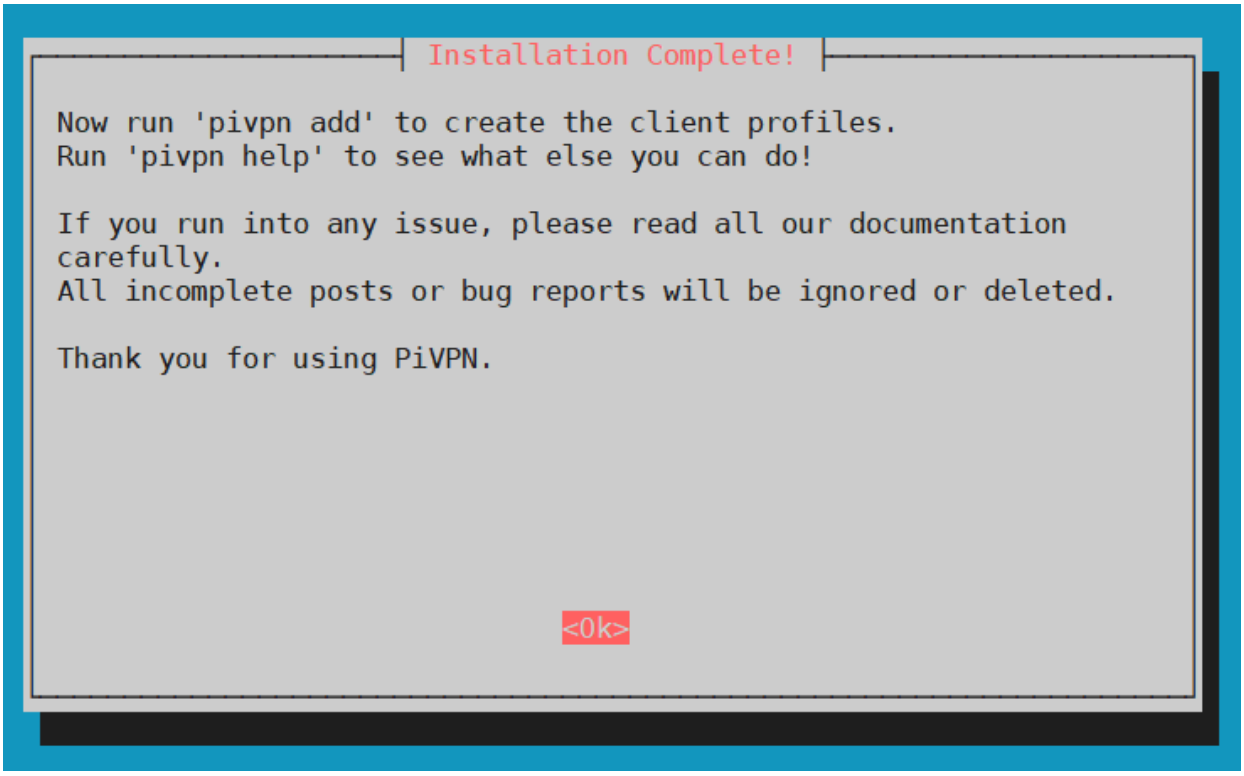
Open the terminal and run the following command

```
curl -L https://install.pivpn.io | bash
```

Then it will prompt with a dialog box and will be asked with a few questions on setting up the OpenVPN server. Here we will choose the default settings because it is enough to make the server up and running.



Installer Start Screen



Installation successful window. Now reboot Raspberry Pi

Now we have successfully installed the OpenVPN server on Raspberry Pi. Check whether it's running by entering the following command.

```
sudo service openvpn status
```

If we see the messages as below, the installation is successful.

```
pi@raspberrypi:~ $ sudo service openvpn status
● openvpn.service - OpenVPN service
   Loaded: loaded (/lib/systemd/system/openvpn.service; enabled; vendor preset: enabled)
   Active: active (exited) since Thu 2020-07-02 09:25:16 BST; 10min ago
  Process: 454 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 454 (code=exited, status=0/SUCCESS)

Jul 02 09:25:15 raspberrypi systemd[1]: Starting OpenVPN service...
Jul 02 09:25:16 raspberrypi systemd[1]: Started OpenVPN service.
```

Port Forwarding

Even though we have set up the OpenVPN server on the Raspberry Pi, it's not possible to access this server remotely because the port we have configured before is not open to access from outside networks. In order to give access to this server, we

need to open that port on the router which is called port forwarding. However, different routers have different methods to set up port forwarding. We can check the router model and search on how to port forward on that specific router.

Creating a client ovpn profile

Now that the server is running, we need to configure the clients which will be connecting to the server. In order to do this, we need to generate an ovpn profile for each and every client that will be accessing server.

Run the following command to add a client

```
pivpn add
```

Then proceed with entering a client name, number of days the certificate would last, and a password

```
pi@raspberrypi:~ $ pivpn add
::: Create a client ovpn profile, optional nopass
:::
::: Usage: pivpn <-a|add> [-n|--name <arg>] [-p|--password <arg>]|[nopass] [-d|--days <number>]
:::
::: Commands:
:::  [none]               Interactive mode
:::  nopass               Create a client without a password
:::  -n,--name            Name for the Client (default: 'raspberrypi')
:::  -p,--password        Password for the Client (no default)
:::  -d,--days            Expire the certificate after specified number of days (default: 1080)
:::  -b,--bitwarden       Create and save a client through Bitwarden
:::  -i,--iOS             Generate a certificate that leverages iOS keychain
:::  -o,--ovpn            Regenerate a .ovpn config file for an existing client
:::  -h,--help            Show this help dialog

Enter a Name for the Client:  User1
How many days should the certificate last?  1080
Enter the password for the client:
Enter the password again to verify:
```

Creating User1 OVPN profile

```
============================================================
Done! User1.ovpn successfully created!
User1.ovpn was copied to:
  /home/pi/ovpns
for easy transfer. Please use this profile only on one
device and create additional profiles for other devices.
============================================================
```

Successfully added User1

However, if we don’t want to specify a password, we can type the following

```
pivpn add nopass
```

To list all valid and revoked certificates, type the following

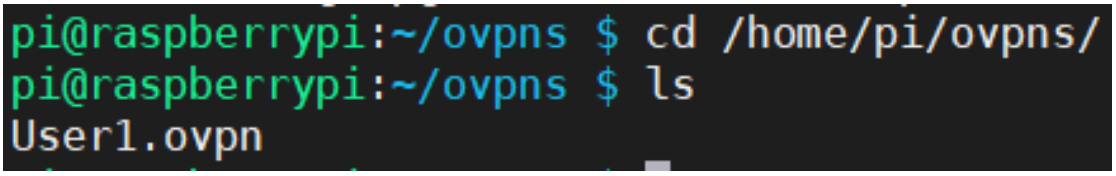
```
pivpn list
```

To revoke a client ovpn profile, type the following

```
pivpn revoke
```

Connecting Clients

When client profiles are added, they get stored under **/home/user/ovpns** directory.



Now that we have a .ovpn file for device, we need to install the necessary software on the client in order to use this file. **OpenVPN Connect** is the recommended software to use on all iOS, Android, macOS, Linux and Windows devices.

IDS – OpenCanary

Command:

```
git clone https://github.com/thinkst/opencanary
cd opencanary
sudo python3 setup.py install
```

Install network add-ons

Install pcapy and scapy:

```
sudo pip3 install scapy pcapy
sudo cp ./build/scripts-3.7/opencanary.tac /usr/local/bin/opencanary.tac
```

Next, run the following command to create a sample config file to the canary to edit:

```
opencanaryd --copyconfig
```

We will see a message saying something like:

```
A sample config file is ready (/etc/opencanaryd/opencanary.conf)
```

Then finally, run canary with the following:

```
opencanaryd --start
```

Install Samba

For OpenCanary honey pot to mimic a windows fileserver, we'll need to enable the SMB protocal (samba):

```
sudo apt install samba samba-common-bin
```

At one point, we will be presented with an ASCII GUI interface asking "**Modify smb.conf to use WINS settings from DHCP?**", answer no.

Rename the smb configuration file (so we can always rollback to the original):

```
sudo mv /etc/samba/smb.conf /etc/samba/smb.conf_backup
```

Create a new configuration file:

```
sudo nano  /etc/samba/smb.conf
```

Paste a configuration for SMB, I use something like the following:

```
[global]
workgroup = OFFICVLAN
server string = Synology Backup
netbios name = SYNOLOGY
dns proxy = no
```

```
log file = /var/log/samba/log.all
log level = 0
vfs object = full_audit
full_audit:prefix = %U|%I|%i|%m|%S|%L|%R|%a|%T|%D
full_audit:success = pread
full_audit:failure = none
full_audit:facility = local7
full_audit:priority = notice
max log size = 100
panic action = /usr/share/samba/panic-action %d
#samba 4
server role = standalone server
#samba 3
#security = user
passdb backend = tdbsam
obey pam restrictions = yes
unix password sync = no
map to guest = bad user
usershare allow guests = yes
[myshare]
comment = Local Backup
path = /home/backups
guest ok = yes
read only = yes
browseable = yes
```

There is currently a quirk of the OpenCanary where SMB printer sharing can self-trigger alerts, resulting in something like the following alert (notice the source and destination IP are both 127.0.0.1):

{"dst_host": "127.0.0.1", "dst_port": "631", "local_time": "2021-09-28 10:45:24.628126", "local_time_adjusted": "2021-09-28 11:45:24.628358", "logdata": {"DF": "", "ID": "29354", "IN": "lo", "LEN": "60", "MAC": "00:00:00:00:00:00:00:00:00:00:00:00:08:00", "OUT": "", "PREC": "0x00", "PROTO": "TCP", "RES": "0x00", "SYN": "", "TOS": "0x00", "TTL": "64", "URGP": "0", "WINDOW": "65495"}, "logtype": 5001, "node_id": "opencanary-1", "src_host": "127.0.0.1", "src_port": "57366", "utc_time": "2021-09-28 10:45:24.628289"}

At the time of writing, this can be solved by installing CUPS (Common UNIX Printing System) though this is likely to be resolved in future versions of OpenCanary. To install CUPS:

```
sudo apt install cups
```

Email Notifications

One of the most common means of getting notifications from a remote OpenCanary is via email, the configuration for which is in the /etc/opencanaryd/opencanary.conf file. This uses the format below (add this to the handlers section):

```
"SMTP": {
"class": "logging.handlers.SMTPHandler",
"mailhost": ["smtp.gmail.com", 587],
"fromaddr": "johndoe@gmail.com",
"toaddrs" : ["securityalerts@bobmckay.com"],
"subject" : "OpenCanary Alert at home!",
"credentials" : ["johndoe@gmail.com", "YOURAPPLICATIONPASSWORD"],
"secure" : []
}
```

At the time of writing, both Office 365 and Gmail require the use of an application password for SMTP connections

IV. TESTING

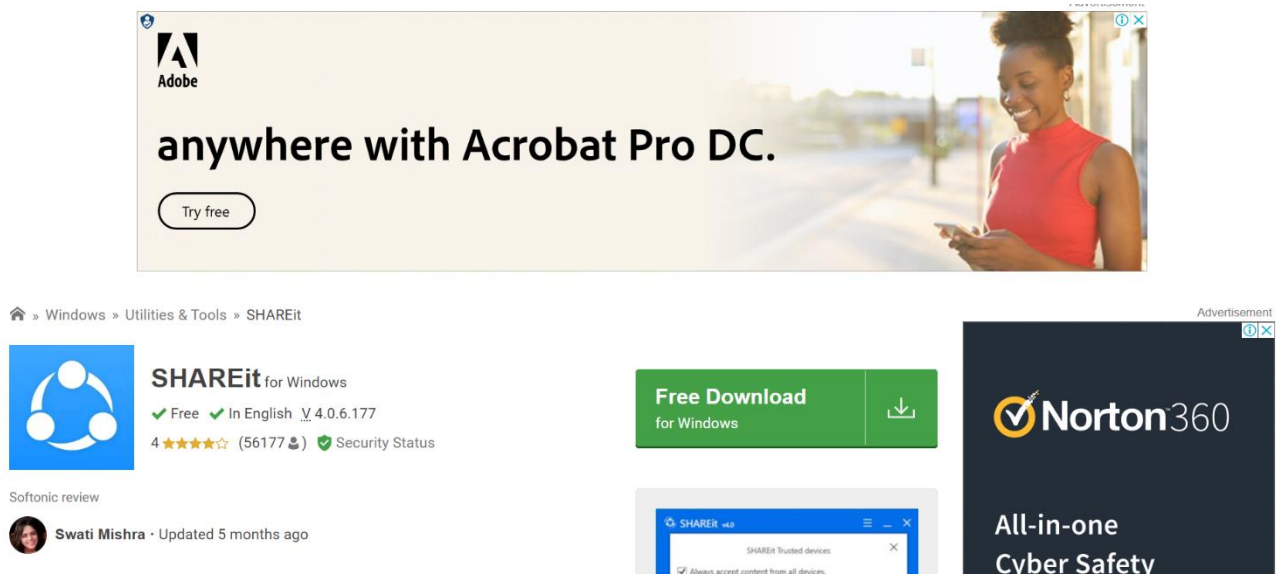
Evaluation strategy

Testing is done in three aspects:

Pi-Hole effect: The testing was conducted on softonic website as it displays many advertisements.

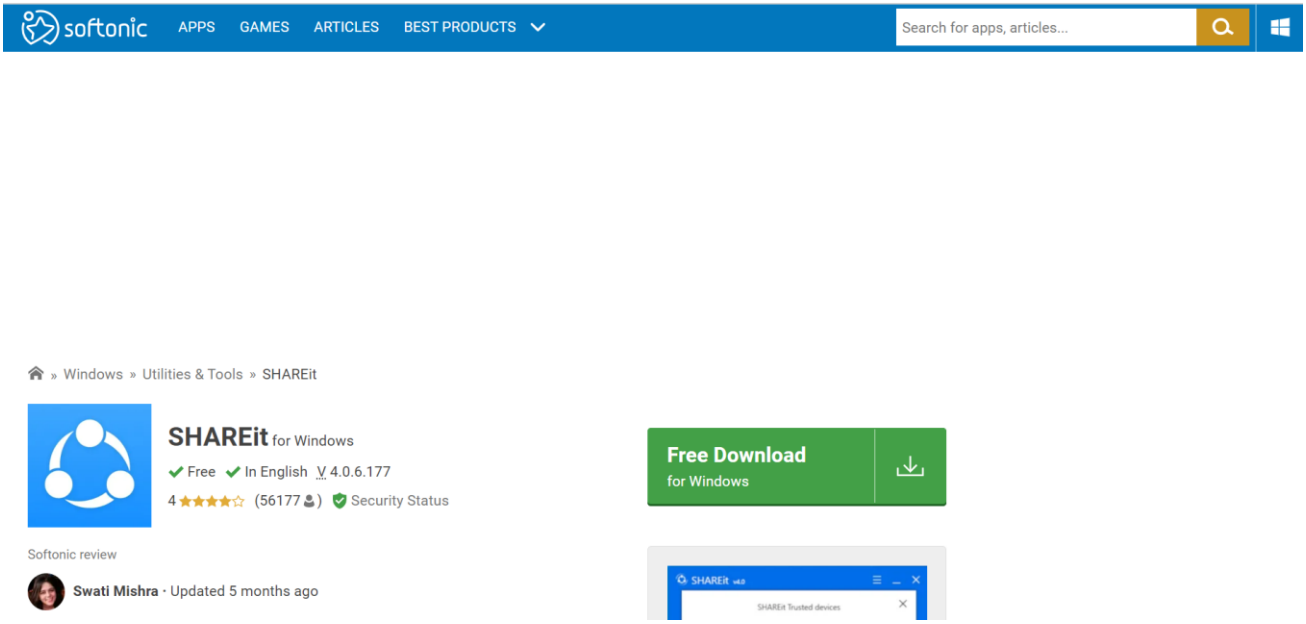
Without Pi-Hole:

Below figure shows the snapshot of website containing several advertisements.

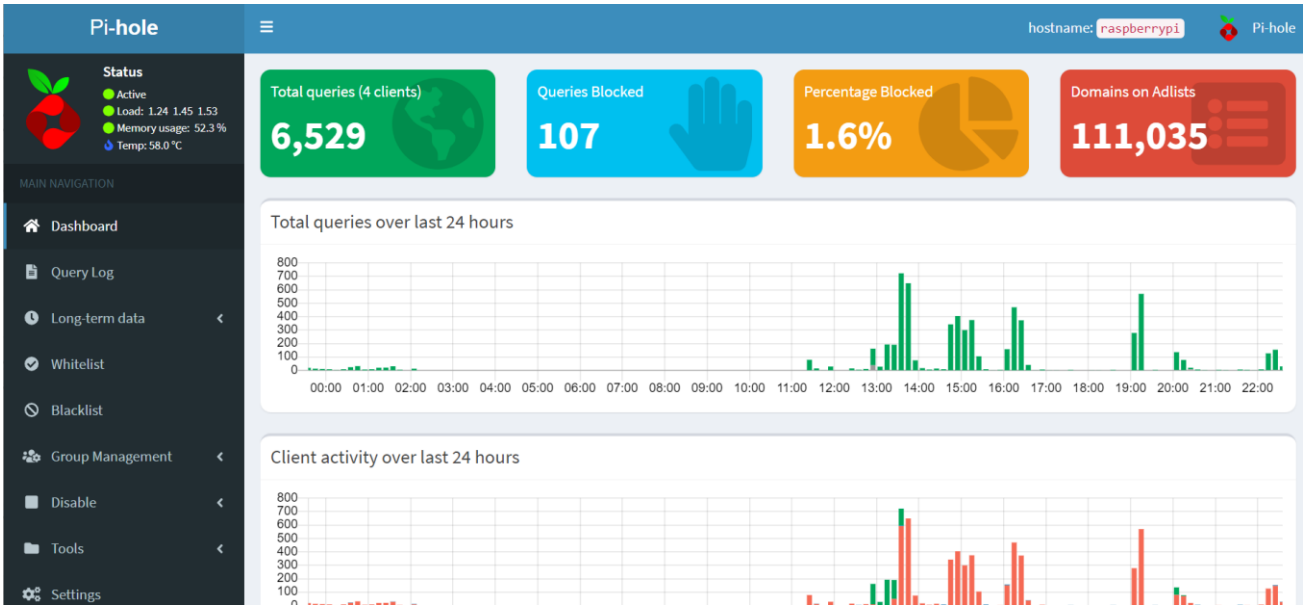


With Pi-Hole:

Below snapshot shows the website with Pi-Hole on. Here the advertisements are not only blocked but the overall loading time of the page is faster.



The admin panel shows total queries and number of queries blocked.

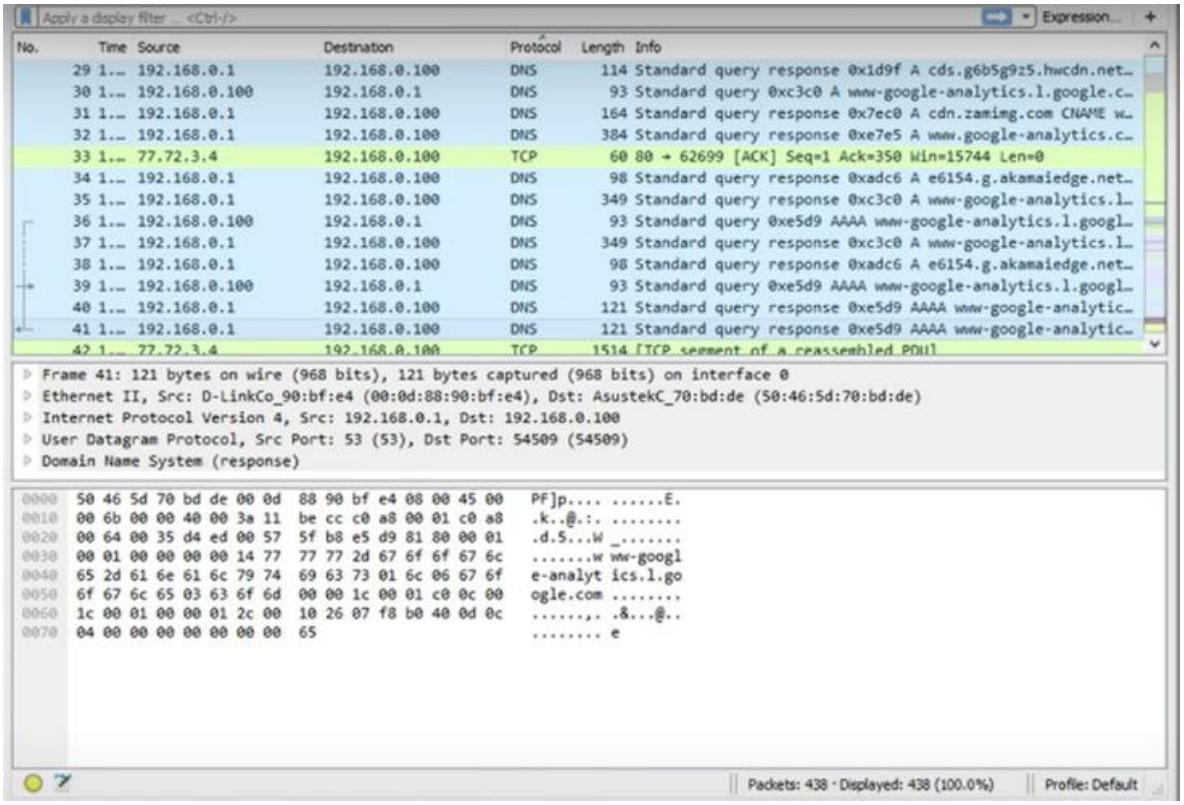


The below figure shows the blocked queries for advertisement.

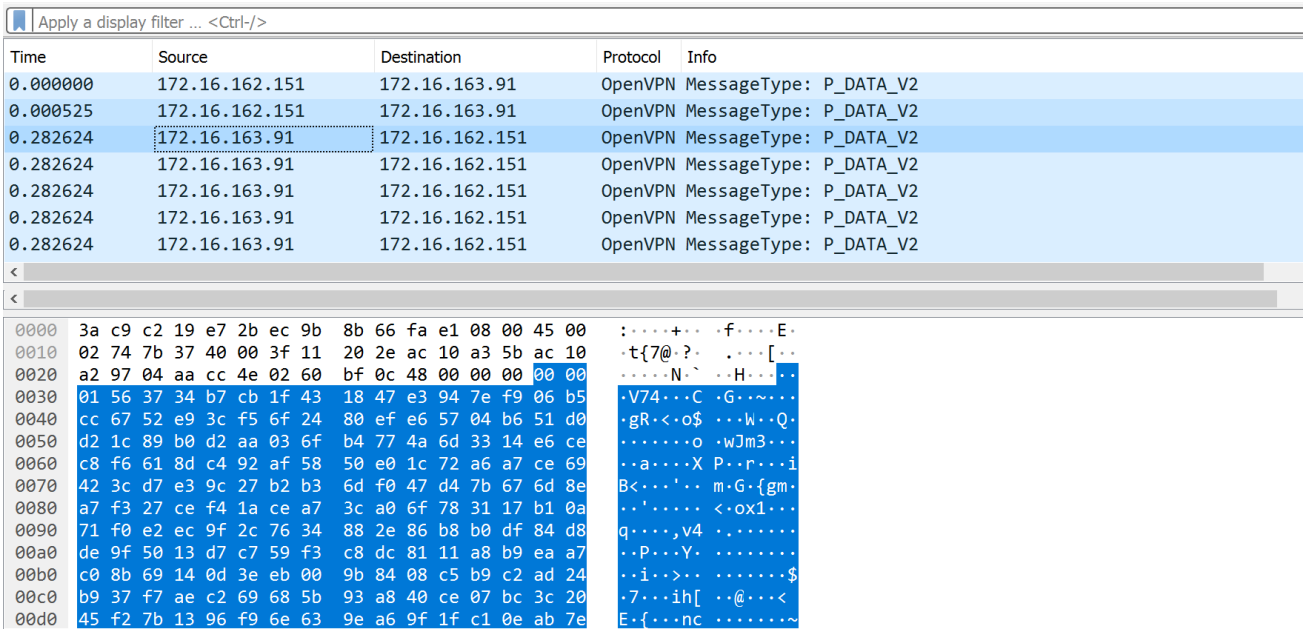
2022-04-24 14:11:46	A	browser.pipe.aria.microsoft.com	client1.pivpn	Blocked (gravity)	IP (0.2ms)	Whitelist
2022-04-24 14:11:43	A	browser.pipe.aria.microsoft.com	client1.pivpn	Blocked (gravity)	IP (0.2ms)	Whitelist
2022-04-24 14:05:34	A	nexusrules.officeapps.live.com	client1.pivpn	Blocked (gravity)	IP (0.1ms)	Whitelist
2022-04-24 14:05:07	A	adservice.google.com	client1.pivpn	Blocked (gravity)	IP (0.1ms)	Whitelist
2022-04-24 14:04:47	A	targeting.api.drift.com	client1.pivpn	Blocked (gravity)	IP (0.2ms)	Whitelist
2022-04-24 14:04:47	A	metrics.api.drift.com	client1.pivpn	Blocked (gravity)	IP (0.2ms)	Whitelist
2022-04-24 14:04:47	A	js.driftt.com	client1.pivpn	Blocked (gravity)	IP (0.2ms)	Whitelist
2022-04-24 14:04:47	A	dev.visualwebsiteoptimizer.com	client1.pivpn	Blocked (gravity)	IP (0.2ms)	Whitelist
2022-04-24 14:04:40	A	targeting.api.drift.com	client1.pivpn	Blocked (gravity)	IP (0.2ms)	Whitelist
2022-04-24 14:04:40	A	js.driftt.com	client1.pivpn	Blocked (gravity)	IP (0.2ms)	Whitelist
2022-04-24 14:04:40	A	metrics.api.drift.com	client1.pivpn	Blocked (gravity)	IP (0.2ms)	Whitelist
2022-04-24 14:04:40	A	www.googletagmanager.com	client1.pivpn	Blocked (gravity)	IP (0.1ms)	Whitelist

OpenVPN Anonymity Testing:

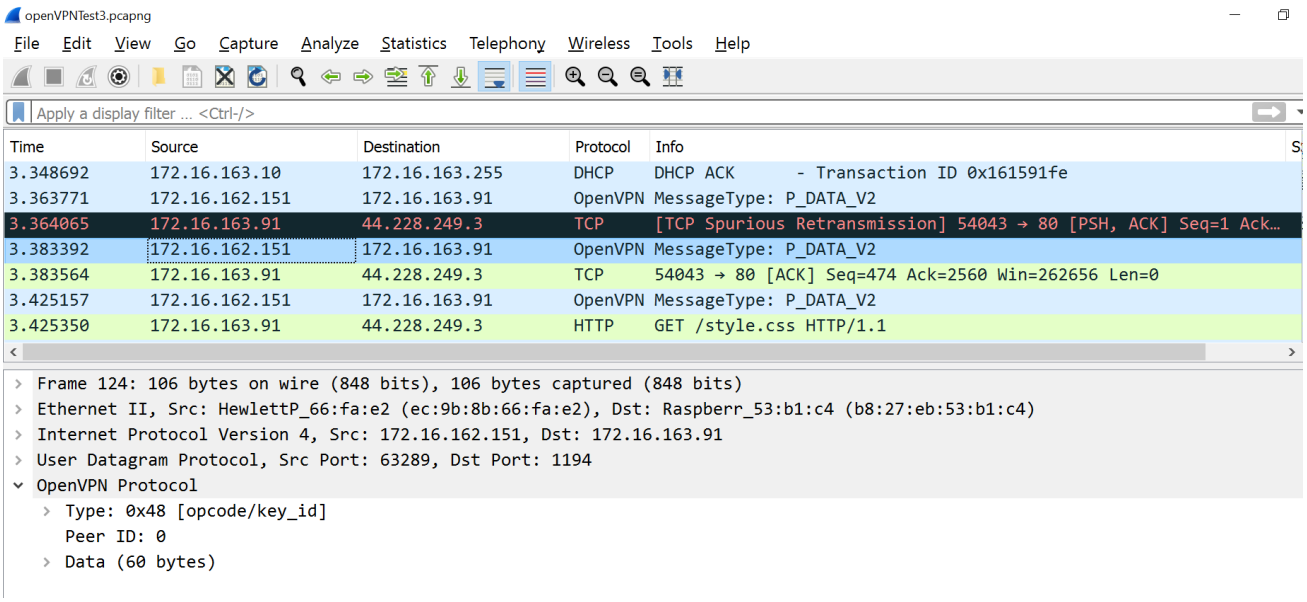
When we don't use OpenVPN, the traces shows that the data is not encrypted.



When we use OpenVPN, the Data from client connected to OpenVPN server is sent in encrypted form from client to the server.



The traces shows that the connected client when sends the packet to the server, after leaving the network, the source ip is changed to the VPN server ip hence for the other (outside the network),the client ip is hidden and it seems like the request is coming from the VPN server.



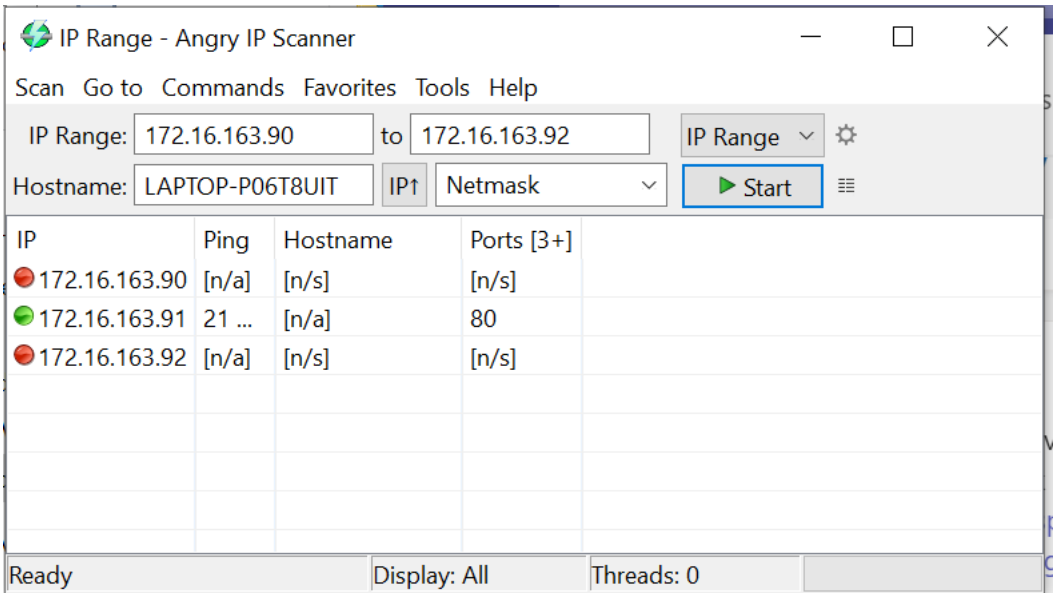
IDS Testing:

First, we need to configure the opencanary.conf file and start the server.

Enabling port scanning detection

```
"portscan.enabled": true,
"portscan.ignore_localhost": true,
"portscan.logfile":"/var/log/kern.log",
```

Launching the port scanning attack from any other machine using **Angry IP scanner**.



On successful port scanning attack, we get corresponding logs in kern.log file as shown below.

```
Apr 24 18:38:20 raspberrypi kernel: [12283.282114] canaryfw: IN=wlan0 OUT= MAC=b8:27:eb:53:b1:c4:ec:9b:8b:66:fa:e2:08:00 SRC=172.19.124.77 DST=172.16.163.91 LEN=52 TOS=0x00
 PREC=0x00 TTL=127 ID=24107 DF PROTO=TCP SPT=54908 DPT=80 WINDOW=32 RES=0x00 SYN URGP=0
Apr 24 18:38:20 raspberrypi kernel: [12283.293615] canaryfw: IN=wlan0 OUT= MAC=b8:27:eb:53:b1:c4:ec:9b:8b:66:fa:e2:08:00 SRC=172.19.124.77 DST=172.16.163.91 LEN=52 TOS=0x00
 PREC=0x00 TTL=127 ID=24110 DF PROTO=TCP SPT=54909 DPT=443 WINDOW=32 RES=0x00 SYN URGP=0
Apr 24 18:38:20 raspberrypi kernel: [12283.410089] canaryfw: IN=wlan0 OUT= MAC=b8:27:eb:53:b1:c4:ec:9b:8b:66:fa:e2:08:00 SRC=172.19.124.77 DST=172.16.163.91 LEN=52 TOS=0x00
 PREC=0x00 TTL=127 ID=24111 DF PROTO=TCP SPT=54910 DPT=8080 WINDOW=32 RES=0x00 SYN URGP=0
```

Hence we are able to detect the attack by implementing OpenCanary IDS in our system.

V. RESULTS

Latency and RTT Calculation

METHOD 1: USING PING COMMAND

Without VPiSec:

```
D:\Nilesh\Study\Assignements\TiN\Project>ping 216.58.196.174

Pinging 216.58.196.174 with 32 bytes of data:
Reply from 216.58.196.174: bytes=32 time=22ms TTL=56
Reply from 216.58.196.174: bytes=32 time=35ms TTL=56
Reply from 216.58.196.174: bytes=32 time=22ms TTL=56
Reply from 216.58.196.174: bytes=32 time=20ms TTL=56

Ping statistics for 216.58.196.174:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 20ms, Maximum = 35ms, Average = 24ms
```

With VPiSec:

```
D:\Nilesh\Study\Assignements\TiN\Project>ping 216.58.196.174

Pinging 216.58.196.174 with 32 bytes of data:
Reply from 216.58.196.174: bytes=32 time=45ms TTL=55
Reply from 216.58.196.174: bytes=32 time=68ms TTL=55
Reply from 216.58.196.174: bytes=32 time=30ms TTL=55
Reply from 216.58.196.174: bytes=32 time=57ms TTL=55

Ping statistics for 216.58.196.174:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 30ms, Maximum = 68ms, Average = 50ms
```

PING Request	Without VPiSec (RTT in ms)	With VPiSec (RTT in ms)
1	22	45
2	35	68
3	22	30
4	20	57
Average RTT(ms)	24	50

The average RTT of with VPiSec is 2 times that of without VPiSec
On average 26 ms of latency is introduced due to implementation of VPiSec.

Latency Measurement between Links (Client to VPN server to Server)

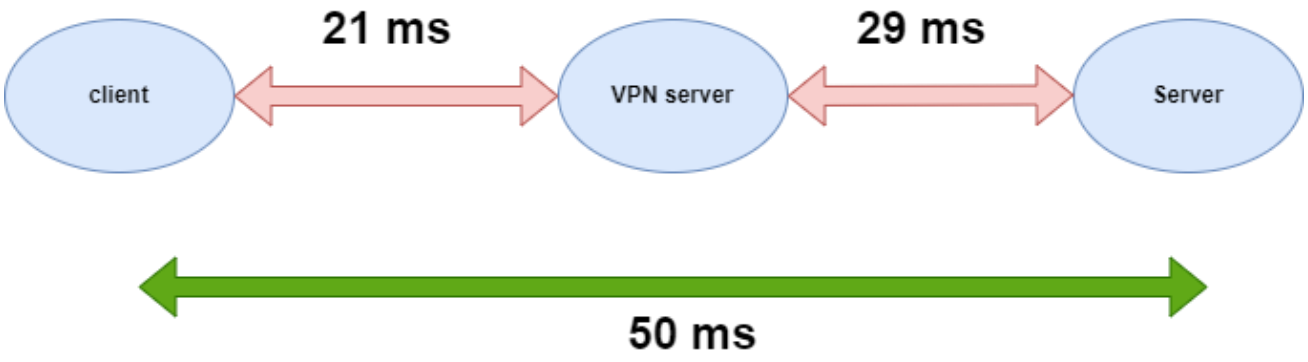
Packet trace at VPN server during PING.

ip.dst == 216.58.196.174 or ip.src == 216.58.196.174

Time	Source	Destination	Protocol	Info	SSID
0.669145	172.16.163.91	216.58.196.174	ICMP	Echo (ping) request id=0x0001, seq=5572/50197, ttl=127 (repl…	
0.695428	216.58.196.174	172.16.163.91	ICMP	Echo (ping) reply id=0x0001, seq=5572/50197, ttl=56 (reque…	
1.693046	172.16.163.91	216.58.196.174	ICMP	Echo (ping) request id=0x0001, seq=5573/50453, ttl=127 (repl…	
1.744185	216.58.196.174	172.16.163.91	ICMP	Echo (ping) reply id=0x0001, seq=5573/50453, ttl=56 (reque…	
2.707890	172.16.163.91	216.58.196.174	ICMP	Echo (ping) request id=0x0001, seq=5574/50709, ttl=127 (repl…	
2.726784	216.58.196.174	172.16.163.91	ICMP	Echo (ping) reply id=0x0001, seq=5574/50709, ttl=56 (reque…	
3.738909	172.16.163.91	216.58.196.174	ICMP	Echo (ping) request id=0x0001, seq=5575/50965, ttl=127 (repl…	
3.761457	216.58.196.174	172.16.163.91	ICMP	Echo (ping) reply id=0x0001, seq=5575/50965, ttl=56 (reque…	

PING Request	Client to client RTT	VPN server to VPN server RTT	RTT introduced by VPiSec
1	45	26	19
2	68	51	17
3	30	18	12
4	57	22	35
Average RTT(ms)	50	29	21

On average 21 ms of latency is introduced due to VPiSec implementation. Pictorial represented as below.



METHOD 2: (Using Python Script)

Without VPiSec

```
US\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.python-2022.4
.1\pythonFiles\lib\python\debugpy\launcher' '61129' '--' 'd:\Nilesh\Study\Assignements\TiN\Project\RTT.py'
Time taken for http://www.amazon.com 0.7896888256072998
PS D:\Nilesh\Study\Assignements\TiN\Project>  d:; cd 'd:\Nilesh\Study\Assignements\TiN\Project'; & 'C:\Users\AS
US\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.python-2022.4
.1\pythonFiles\lib\python\debugpy\launcher' '61146' '--' 'd:\Nilesh\Study\Assignements\TiN\Project\RTT.py'
Time taken for http://www.facebook.com 1.3309130668640137
PS D:\Nilesh\Study\Assignements\TiN\Project>  d:; cd 'd:\Nilesh\Study\Assignements\TiN\Project'; & 'C:\Users\AS
US\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.python-2022.4
.1\pythonFiles\lib\python\debugpy\launcher' '61155' '--' 'd:\Nilesh\Study\Assignements\TiN\Project\RTT.py'
Time taken for http://www.google.com 0.18652868270874023
PS D:\Nilesh\Study\Assignements\TiN\Project>
```

With VPiSec:

```
PS D:\Nilesh\Study\Assignements\TiN\Project>  d:; cd 'd:\Nilesh\Study\Assignements\TiN\Project'; & 'C:\Users\AS
PS D:\Nilesh\Study\Assignements\TiN\Project>  d:; cd 'd:\Nilesh\Study\Assignements\TiN\Project'; & 'C:\Users\AS
Time taken for http://www.amazon.com 2.136021852493286
PS D:\Nilesh\Study\Assignements\TiN\Project>  d:; cd 'd:\Nilesh\Study\Assignements\TiN\Project'; & 'C:\Users\AS
US\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.python-2022.4
.1\pythonFiles\lib\python\debugpy\launcher' '55992' '--' 'd:\Nilesh\Study\Assignements\TiN\Project\RTT.py'
Time taken for http://www.facebook.com 2.3179850578308105
PS D:\Nilesh\Study\Assignements\TiN\Project>  d:; cd 'd:\Nilesh\Study\Assignements\TiN\Project'; & 'C:\Users\AS
US\AppData\Local\Programs\Python\Python37\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.python-2022.4
.1\pythonFiles\lib\python\debugpy\launcher' '61095' '--' 'd:\Nilesh\Study\Assignements\TiN\Project\RTT.py'
Time taken for http://www.google.com 2.213104486465454
```

Observation: We have requested three websites and observed the RTT of both in the above snapshot.

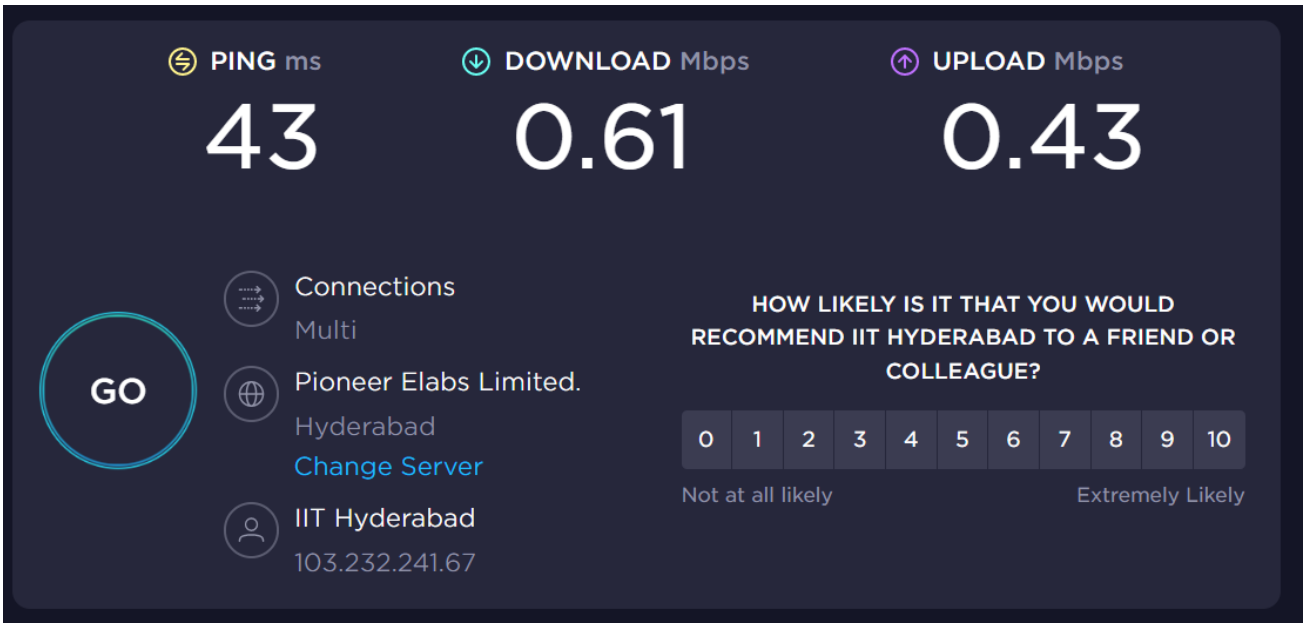
	Without VPiSec (RTT in sec)	With VPiSec (RTT in sec)
Amazon	0.789	2.13
Facebook	1.33	2.31
Google	0.18	2.21
Average RTT(sec)	0.766	2.216

The average RTT of with VPiSec is 2.89 times that of without VPiSec

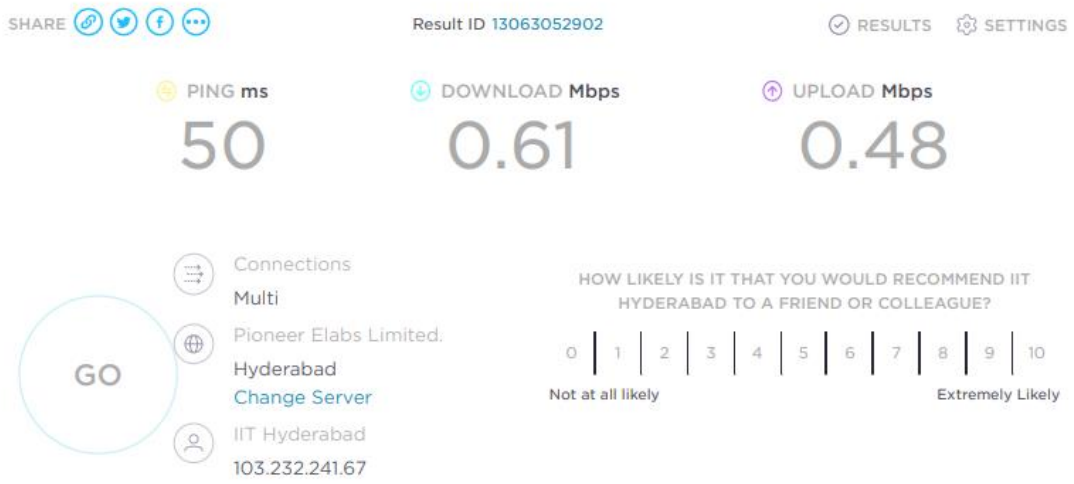
On average 1.45 sec of latency is introduced due to implementation of VPiSec.

Bandwidth Utilization:

Without VPiSec:



With VPiSec:



Here, the upload speed we get with VPiSec is better and there is insignificant change in download speed and latency.

Key results

- It is hiding the user's data and provide anonymity.
- It is be able to block all advertisements.
- We notice increase in Upload Speed and the download speed is almost the same.
- The ping is slightly more but the difference is insignificant.

VI. CONCLUSIONS

The configuration for VPiSec is straightforward, as it was created with the goal of allowing even inexperienced users to utilise and use it to secure their network. The user establishes a connection with VPiSec, which serves as a middleman between the user and the Internet. The experiments were carried out on the VPiSec in order to test the functionality of the OpenVPN, Pi-hole, and OSSEC IPS. Advertisements had successfully blocked programmes as well as websites, according to the Pi-hole test. If a new site tracker query is made by advertisement services, the user must update the blocklist to ensure that Pi-hole can continue to block advertisements. VPiSec assists internet users in protecting their network activities. This project proposes enhancing a programme that allows Pi-hole to automatically update its database. Additionally, it updates the router settings for the IP table to redirect all traffic that enters the network to the VPiSec.

VII. WORK PLAN

TASK	STUDENT NAME	DUE DATE
Read about OpenVPN deployment	Nilesh Kale	7/04/2022
Read about connection to raspberry pi using ssh	Nilesh kale, Amit Kumar	7/04/2022
Install raspbian on raspberry pi	Devang Dubey	8/04/2022
Read about IDS	Amit Kumar	09/04/2022
Read how to connect to OpenVPN using client software	Nilesh Kale	10/04/2022
Making a connection between VPN and Raspberry-pi	Nilesh Kale, Amit Kumar	12/4/2022
Installing Pi hole in raspberry pi	Devang Dubey	14/4/2022
Installed OpenCanary IDS	Amit Kumar	17/4/2022
Tested port scanning on IDS	Devang Dubey	19/4/2022
Tested the pi-hole effect on internet speed	Nilesh Kale	19/4/2022
Tested OpenVPN effect on internet speed,	Devang Dubey, Amit Kumar	22/4/2022
Measured latency difference	Nilesh Kale, Devang Dubey	22/4/2022
Link to Link latency measurement	Nilesh Kale, Devang Dubey	23/4/2022
Detailed documentation	Devang Dubey, Nilesh Kale, Amit Kumar	23/4/2022