UNIVERSITY OF ALBERTA

MINT 709

Project Report on

Smart home devices and their security vulnerabilities

Submitted By:

Himank Sarna

In partial fulfilment for the award of the degree

Master of Science in Internetworking

(From University of Alberta)

Under the guidance of

Professor Leonard Rogers

Acknowledgement

I would like to express my sincere gratitude to Professor Leonard Rogers for his mentorship and untiring efforts in guiding and helping me understand and overcome problems that would otherwise have stopped me midway. Along with this, I thank him for always being approachable and guiding me through constant comments and suggestions on the project.

I would also take this opportunity to thank Dr. Mike MacGregor for approving the project so that I can work on it.

Last but not least, I would like to express my heartfelt gratitude to my parents for their support and encouragement at every step.

Finally, I'd like to thank all unnamed people who have willingly helped me out with their abilities.

Abstract

In recent years, the popularity of IoT devices has been rising and find their application in every industry and homes. It is estimated that there will be over 30 billion IoT devices by the year 2020. With the increased popularity of IoT devices, it has led to many security compromises making devices prone to hacking and exposing sensitive data.

The primary purpose of this project is to make IoT devices more secure by connecting them with an advanced pfSense firewall.

Generally, the IoT devices are connected with the same network as the other devices, and they become a potential attack point. In this project, the IoT devices are connected to a dedicated wireless network which restricts the traffic coming from other networks.

pfSense maintains a stateful firewall with rules to allow and block traffic coming into the network from unknown sources and hosts, maintaining a log of incoming malicious packets.

The static ARP entry allows trusted devices to connect to the network and blocks unknown devices that attempt to establish a connection. NAT features port forwarding so that any incoming connection is redirected to a specific IP address so that other devices in the network do not get compromised.

pfSense features an IDS/IPS that increases the security level of the network. It monitors traffic, detects and blocks malicious packets from the internet to enter into the network.

UUID is a Universally Unique Identifier which identifies any device on the internet. Finally, in this project, python code is programmed which fetches the IP address, MAC address and UUID of the device, and could be implemented with pfSense, adding UUID as a new connection parameter, and further securing the connection.

Index

1. Introduction

pfSense is a trusted open source network security software which can be installed on any physical computer to make a dedicated firewall for a network. Generally, all the devices are connected to the gateway router, which exposes devices within the network to external attacks, like UPnP protocol is enabled on internet facing ports that allow outsiders to access network inside; and, man-in-the-middle attack. The implementation of the pfSense firewall/ router is divided into several sections.

The first section refers to the configuration of pfSense on bare metal CPU with AMD architecture. pfSense image is installed on the CPU which makes it boot into pfSense console every time the CPU restarts.

The second section explains the wireless interface for connecting IoT devices with the firewall/ router. By default, pfSense has two wired interfaces for LAN and WAN. pfSense can be made wireless compatible either by connecting a wireless network card or a hotspot.

VLAN and DHCP section talks about separating the main network into different networks so that IoT devices can connect to their own wireless network. Creating VLANs also secure the network by not allowing the data from other networks. DHCP provides the IP address to the devices connecting to the network, and by reducing the network subnet mask, unauthorized devices do not get an IP address, and the connection gets refused.

pfSense maintains rule-based advanced NAT table, blocking random sessions initiated by unknown devices. It also scrambles the source port adding another security feature. NAT section explains about rules in LAN and WAN and configuring port forwarding to enable an RDP session into the network.

A firewall monitors incoming and outgoing traffic and permits or blocks packets based on the assigned rules. The firewall section talks about configuring firewall rules and firewall aliases. It also gives details about firewall packages such as pfBlockerNG and Squid that extend the pfSense firewall's capabilities based on certain known blacklisted domains and setting up ACLs.

IPS and IDS are integral from a security standpoint. IDS is the process of monitoring the events in the network, detecting and identifying log violations so that IPS can act on the malicious packets detected and stop them from entering into the network. The IPS/IDS section highlights an open source IDS and IPS Snort which detects and suppresses exploits and malware through rulesets.

Finally, the last section talks about UUID, which stands for Universally Unique Identifiers. It is a 128-bit long hexadecimal string of characters which is unique for every device over the internet. A python program is used to find the IP and MAC address of the device, and UUID as a new connection parameter. This program with additional pfSense libraries can be incorporated with the pfSense firewall to establish a connection with a device based on its UUID.

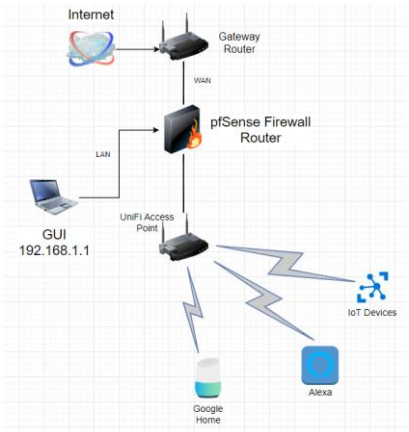


Figure 1 pfSense firewall block diagram

The above figure is the block diagram of pfSense firewall router implementation, in which all the IoT devices get connected wirelessly to the UniFi access point, that is connected to the pfSense firewall.

The web GUI is accessed through a PC on the local LAN.

Any device can be connected to pfSense through wired or wireless on the basis of permit and deny firewall rules, and a log is created every time a connection is made.

The WAN interface of pfSense is connected with the gateway router which provides the internet connection to all the devices on different networks at LAN side.

2. Configuring pfSense

pfSense is an open source firewall/router software distribution based on FreeBSD. It is a free, customized software which can be installed on any device with AMD architecture. pfSense is very flexible and adaptable with numerous applications which can be accessed using a web GUI. It provides a lot of features like firewall, routing, IDS, IPS, proxy and content filtering, system security, reporting and monitoring, and many more.

2.1. Change in Architecture

The proposal stated the implementation of pfSense on Raspberry Pi. However, pfsense extends its compatibility only with Netgate ARM-based devices. These devices are already installed with the factory version of pfSense software. On researching more about pfSense's compatibility with ARM-based architecture, it came into light that BSD kernel was not stable with ARM. Raspberry Pi does support FreeBSD (the platform same as pfSense) but running pfSense seemed incompatible. Hence, for implementation of the pfSense firewall router, AMD64 architecture has been used.

2.2. Hardware Specifications

pfSense version 2.4.4 release patch-1 has is installed on a bare metal device with AMD64 architecture. It is a prerequisite for AMD64 device to have two ethernet ports for enabling functionality of LAN and WAN. Several network adapters can be added to support the functioning of pfSense as a wireless Access Point for deploying wireless connectivity. After the pfSense is installed on the hard drive of the system, it boots up with a series of configuration steps in which the interfaces, i.e., WAN, LAN, and OPT are configured with IP address. After assigning the IP address to respective interfaces, it boots up in shell menu with some options.

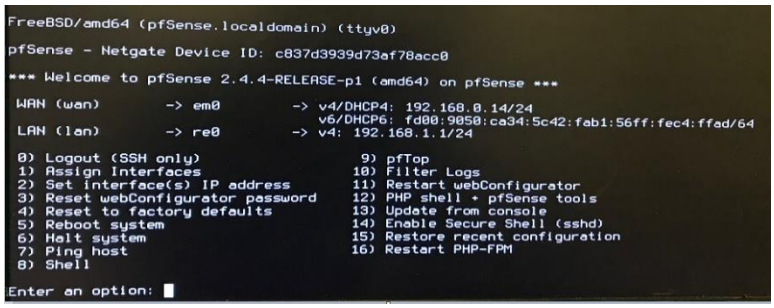


Figure 2 pfSense shell menu

2.3. WebGUI

The LAN interface gives access to web GUI which opens a pfsense web interface through IP address 192.168.1.1.

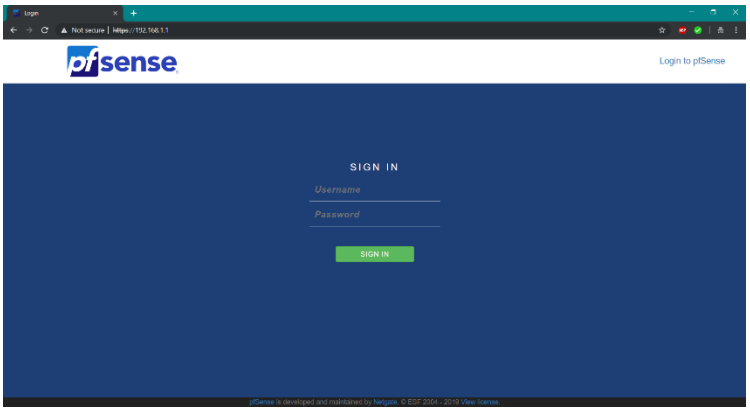


Figure 3 pfSense shell menu

Logging into this web GUI generates a log on the console menu stating the IP address with date and time of login. 192.168.1.100 is the IP address of the device connected on the network through LAN ethernet.

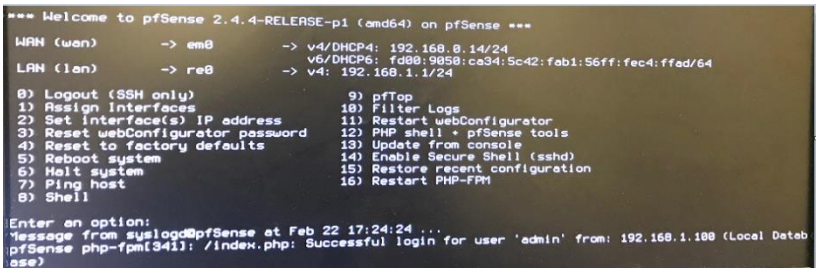


Figure 4 Log creation at login

```
C:\Users\Himank Sarna>tracert google.com

Tracing route to google.com [172.217.3.174]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  Capstone-Pfsense.localdomain [192.168.1.1]
  2     3 ms     2 ms     1 ms  192.168.0.1
  3     *        *        *     Request timed out.
  4    14 ms    12 ms    11 ms  rc3ar-be114-1.ed.shawcable.net [64.59.186.121]
  5    15 ms    90 ms    29 ms  66.163.70.129
  6    38 ms    17 ms    15 ms  rc3no-be6.cg.shawcable.net [66.163.64.69]
  7    34 ms    31 ms    29 ms  rc2wt-be100.wa.shawcable.net [66.163.75.233]
  8    30 ms    42 ms    49 ms  72.14.242.90
  9     *        *        *     Request timed out.
 10    38 ms    34 ms    32 ms  74.125.253.60
 11    30 ms    31 ms    31 ms  108.170.233.157
 12    29 ms    38 ms    27 ms  sea15s11-in-f174.1e100.net [172.217.3.174]

Trace complete.
```

Figure 5 Tracert output with pfSense

The web interface provides access to all the functionalities of pfSense which includes accessing interfaces, system information, firewall capabilities, traffic monitoring, IDS/IPS and other essential features. The dashboard provides easy access to all the functionality of pfSense displaying important log notifications, traffic monitor and status of connections.

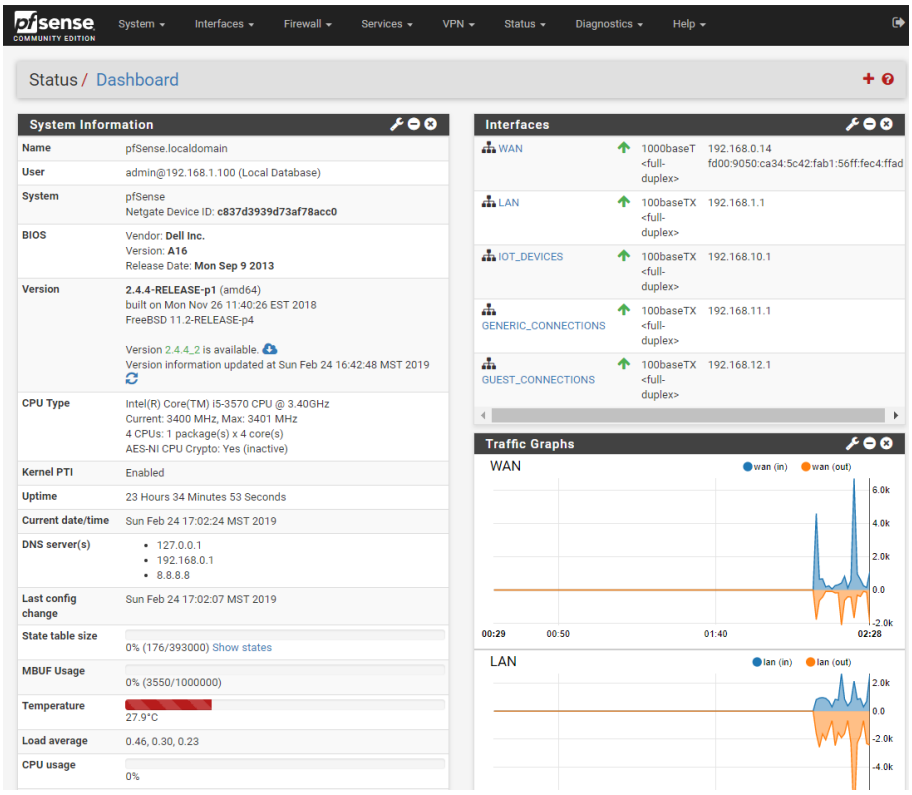


Figure 6 pfSense web GUI dashboard

3. Wireless Configuration

pfSense supports wireless compatibility through the OPT interface. For IoT devices to connect wirelessly to the pfSense firewall router, it is necessary to configure the wireless interface. pfSense supports access point functionality with 802.11n and 802.11ac support. Using a wireless network adapter, IoT devices are connected wirelessly to the firewall hardware, providing firewall security with internet functionality at the same time. The list of physical compatible wireless network interfaces are as follows:

Wireless Interfaces

		Supported Modes of Operation												
Driver Name		hostap	station	monitor	adhoc	adhoc demo	mesh	wds	bgscan	wme	wpa	# Virtual Interfaces	Firmware	Kernel Devices
bwn	Broadcom 802.11n	No	Yes	Yes	No	Yes	No	No	Yes	Yes	Yes	1	net/bwn-firmware-kmod	bwn, siba_bwn, wlan_amrr, firmware
ipw	Intel Pro/Wireless	No	Yes	Yes	Yes	No	No	No	No	No	Yes	1	ipwfw	ipfw, ipfwfw, firmware
iwi	Intel Pro/Wireless	No	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	1	iwifw	iwi, iwifw, pci, firmware
iwn	Intel AGN	No	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	1	iwnfw	iwn, iwnfw, pci, firmware
malo	Marvell wireless	No	Yes	Yes	No	No	No	No	Yes	No	Yes	1	URL in man page	malo, firmware
mwl	Marvell 802.11N	Multi	Multi	Yes	No	No	Yes	Multi	Yes	Yes	Yes	8 per type	mwlfw	mwl, mwlfw, firmware
ral	Ralink	Yes	Multi	Yes	Yes	No	Yes	Multi	Yes	Some	Yes	No Limit	ralfw	ral, ralfw, wlan_amrr, firmware
rum	Ralink USB	Yes	Yes	Yes	Yes	No	No	No	Yes	No	Yes	1		rum, wlan_amrr
run	Ralink USB 802.11n	Multi?	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes	8?	runfw	run, runfw
uath	Atheros USB	No	Yes	Yes	No	No	No	No	Yes	No	Yes	1	uathload	uath
upgt	Conexant/Intersil PrismGT	No	Yes	Yes	No	No	No	No	Yes	No	Yes	1	URL in man page	upgt
ural	Ralink USB	Yes	Yes	Yes	Yes	No	No	No	Yes	No	Yes	1		ural, wlan_amrr
urtw	Realtek USB	No	Yes	Yes	No	No	No	No	Yes	No	Yes	1		urtw
wi	Lucent/Intersil Prism	Yes	Yes	Yes	Yes	No	No	No	No	No	Yes	1		wi
wpi	Intel 3945ABG	No	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	1	wpifw	wpi, wpifw, wlan_amrr, firmware
zyd	ZyDAS	No	Yes	Yes	No	No	No	No	Yes	No	Yes	1		zyd, wlan_amrr
ndis	NDIS miniport													

Figure 7 pfSense supported wireless drivers

Three network adapters: PEX300WN2X2 PCI Express Wireless N Card, USB-AC51 Dual-Band Wireless AC600 Network Adapter, USB 2.0 and TP-Link TL-WN881ND Wireless N300 with plug and play capabilities were tested but did not work. These network cards belonged to 'Atheros' having compatibility with pfsense.

The following is taken from the official Netgate website which states that the compatible network cards may not function with the hardware in some cases.

"Some care is needed when testing your hardware to see if this feature is supported. Some chips will fail to add the additional interface; others may panic and cause a reboot."

Unifi access point has been used to enable wireless connectivity of IoT devices with pfSense firewall. The Unifi access point plugged with an ethernet port of the pfSense device, creates a wireless network of different VLANs.

Figure 8 UniFi Dashboard

With access point enabled, wireless networks with VLAN tagging become functional. This access point enables the wireless accessibility of networks VLAN tags 10, 20, 30 and 40 created in pfSense.

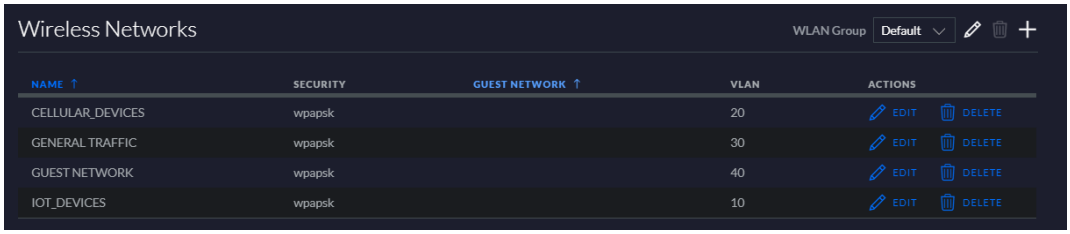


Figure 9 Networks with VLAN tagging

The wireless networks have WPA Enterprise security. Radius Server secures the connection with the access point and clients to a great extent. The client associates to the access point and radius server generates a random 256 bits PMK to encrypt data for the current session.

PMK is unique and session specific for each client, therefore if someone tries to break a PMK, only one session of that client is accessed.

Hiding the SSID of that network makes it less vulnerable for people trying to connect using wireless.

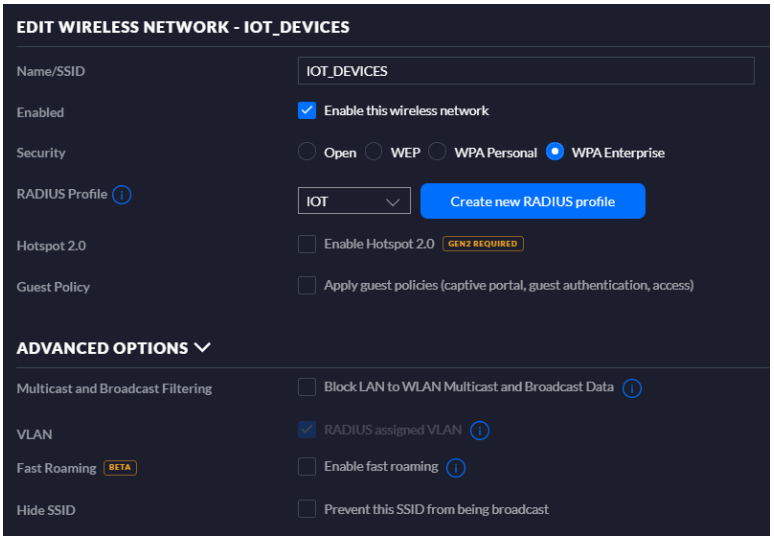


Figure 10 IoT network with WPA Enterprise Security

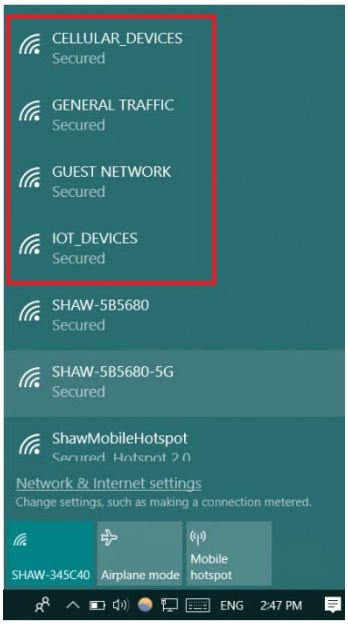


Figure 11 List of available WiFi networks

For security measures, the IoT network can only allow devices to connect to it. DHCP range is starting with 192.168.10.4 to 192.168.10.6. 192.168.10.2 is the static IP address mapped to Google Home using static ARP. If a device connects to the network, a log is generated capturing its MAC address, and the connection request to the device is denied.

NAME	IP ADDRESS	WIFI EXPERIENCE	CONNECTION	AP/PORT	ACTIVITY	ACTIVITY DOWN	ACTIVITY UP	UPTIME
Google-Home	192.168.10.2	0%	IOT_DEVICES	b4:fb:e4:10:1c:e2		72.3 KB	1.2 MB	13m 25s
Himank-Sarna	192.168.10.4	0%	IOT_DEVICES	b4:fb:e4:10:1c:e2		269 KB	263 KB	16m 8s
HSSSTD011	192.168.10.5	0%	IOT_DEVICES	b4:fb:e4:10:1c:e2		9.37 KB	26.2 KB	6m 10s
IPhone	192.168.10.6	0%	IOT_DEVICES	b4:fb:e4:10:1c:e2		120 B	13.9 KB	3m 34s
zain	169.254.228.94	0%	IOT_DEVICES	b4:fb:e4:10:1c:e2		0 B	40.4 KB	48s

Figure 12 IoT network clients

MAC addresses are categorized into Whitelist and Blacklist. Whitelist MAC addresses are the known devices which connect to the network anytime; all the other devices are blacklisted by default and the connection is denied.

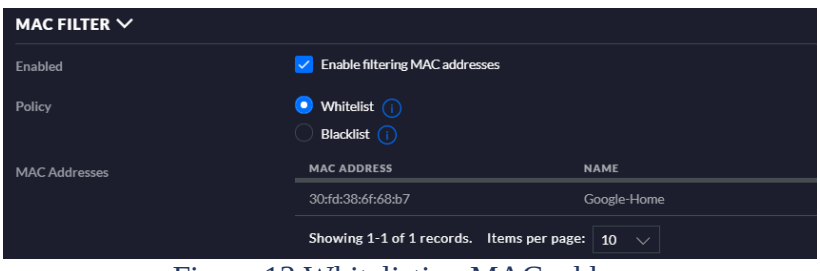


Figure 13 Whitelisting MAC address

4. VLANs and DHCP

VLANs are a logical grouping of a network that divides the LAN into subnetworks. With the help of a VLAN, hosts on a specific network can be isolated from other networks. This provides an additional layer of security in the pfSense firewall as traffic from other networks gets denied.

DHCP is a protocol which allows pfSense to dynamically allocate an IP address from a predefined pool of IP address. In DHCP, services like 'deny unknown clients'/static ARP' and multiple address pool further increase the security in case of unknown hosts.

4.1. Setting up VLANs

pfSense provides the capacity to implement VLANs which are a great way to segment a network and isolate subnetworks. VLANs can be created on any interface providing features and benefits like:

- An additional layer of security by not permitting traffic from one VLAN into another.
- Creating separate networks for different devices to be connected to their network.

The wireless network has been created into four different networks through VLANs, separating traffic from IoT devices from others. Having a crucial need to isolate IoT devices' network, securing them becomes very important to prevent them from being compromised.

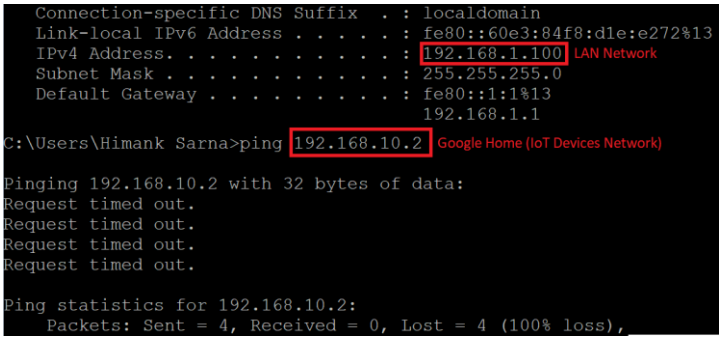


Figure 14 Unsuccessful ping from LAN to IoT Devices network

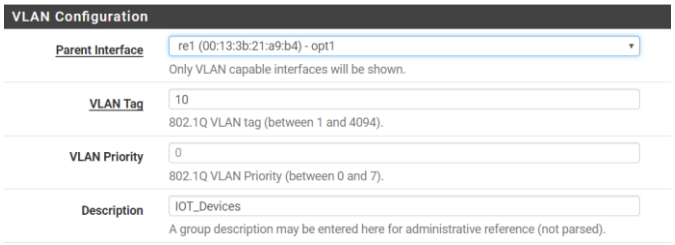
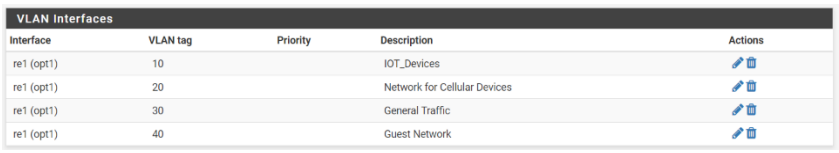


Figure 15 IoT network VLAN configuration



VLAN Interfaces

Interface	VLAN tag	Priority	Description	Actions
re1 (opt1)	10		IOT_Devices	
re1 (opt1)	20		Network for Cellular Devices	
re1 (opt1)	30		General Traffic	
re1 (opt1)	40		Guest Network	

Figure 16 VLAN networks

4.1. DHCP Pool

pfSense router has a DHCP server for dynamically assigning an IP address to various devices in different networks. IoT devices connect to the router wirelessly which leaves them exposed to a rogue DHCP server. To secure the connection, the subnet mask of each network is limited to a certain number of devices, which has the capacity to being increased in the future. This limits the scope of unauthorised devices getting an IP address and hence the connection to that network is denied.

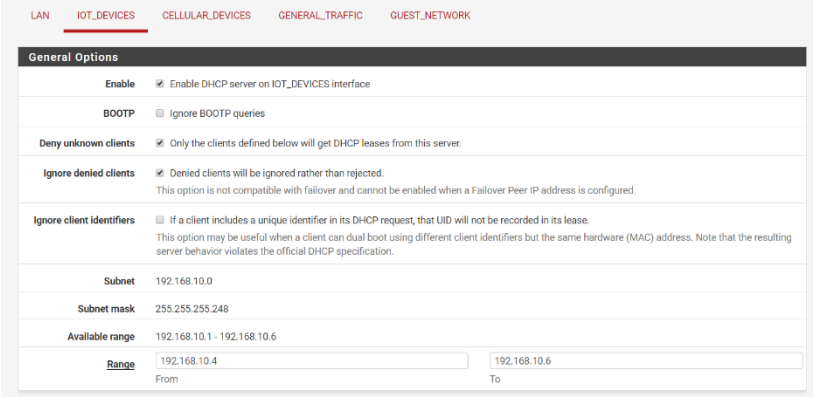


Figure 17 DHCP Pool for IOT_Devices network

'Deny unknown clients' feature provides an additional layer of security. With this feature checked, no clients will get DHCP lease if the client is not defined in 'static DHCP mapping' table. Any incoming request from unknown clients to get connected gets automatically blocked.

It also prevents the risk of MAC spoofing to a certain extent because if any other device is not being given access, it cannot register itself until has the physical access to the device.

Advantages of creating static mapping entry:

- Static ARP entry.
- Mapping of IP address to specific MAC address.

5. NAT

NAT is a process where the firewall assigns a public IP address to the devices inside a private network. NAT is categorised as outbound and inbound where a private IP address is mapped with a single public IP address and vice versa in case of inbound NAT.

What makes it secure is that an outbound connection gets recorded into a translation table and the traffic replying back to that specific connection is allowed back. When a random session gets initiated from outside, it gets blocked.

Automatic outbound NAT translates any internal network subnet to an external WAN IP address.

Automatic Rules:

	Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description
✔	WAN	127.0.0.0/8 ::1/128 192.168.1.0/24 192.168.20.0/24 192.168.10.0/29 192.168.30.0/30 192.168.40.0/28 192.168.50.0/27	*	*	500	WAN address	*	✔	Auto created rule for ISAKMP
✔	WAN	127.0.0.0/8 ::1/128 192.168.1.0/24 192.168.20.0/24 192.168.10.0/29 192.168.30.0/30 192.168.40.0/28 192.168.50.0/27	*	*	*	WAN address	*	⤮	Auto created rule

Figure 18 Automatic outbound NAT rules

Mappings

	Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description	Actions
✔	WAN	127.0.0.0/8	*	*	500 (ISAKMP)	WAN address	*	✔	Auto created rule for ISAKMP - localhost to WAN	
✔	WAN	127.0.0.0/8	*	*	*	WAN address	*	⤮	Auto created rule - localhost to WAN	
✔	WAN	::1/128	*	*	500 (ISAKMP)	WAN address	*	✔	Auto created rule for ISAKMP - localhost to WAN	
✔	WAN	::1/128	*	*	*	WAN address	*	⤮	Auto created rule - localhost to WAN	
✔	WAN	192.168.1.0/24	*	*	500 (ISAKMP)	WAN address	*	✔	Auto created rule for ISAKMP - LAN to WAN	
✔	WAN	192.168.1.0/24	*	*	*	WAN address	*	⤮	Auto created rule - LAN to WAN	
✔	WAN	192.168.20.0/24	*	*	500 (ISAKMP)	WAN address	*	✔	Auto created rule for ISAKMP - WIRELESS to WAN	
✔	WAN	192.168.20.0/24	*	*	*	WAN address	*	⤮	Auto created rule - WIRELESS to WAN	
✔	WAN	192.168.10.0/29	*	*	500 (ISAKMP)	WAN address	*	✔	Auto created rule for ISAKMP - IOT_DEVICES to WAN	
✔	WAN	192.168.10.0/29	*	*	*	WAN address	*	⤮	Auto created rule - IOT_DEVICES to WAN	
✔	WAN	192.168.30.0/30	*	*	500 (ISAKMP)	WAN address	*	✔	Auto created rule for ISAKMP - CELLULAR_DEVICES to WAN	
✔	WAN	192.168.30.0/30	*	*	*	WAN address	*	⤮	Auto created rule - CELLULAR_DEVICES to WAN	
✔	WAN	192.168.40.0/28	*	*	500 (ISAKMP)	WAN address	*	✔	Auto created rule for ISAKMP - GENERAL_TRAFFIC to WAN	
✔	WAN	192.168.40.0/28	*	*	*	WAN address	*	⤮	Auto created rule - GENERAL_TRAFFIC to WAN	
✔	WAN	192.168.50.0/27	*	*	500 (ISAKMP)	WAN address	*	✔	Auto created rule for ISAKMP - GUEST_NETWORK to WAN	

Figure 19 Manual outbound NAT rules

5.1. Port Forwarding

Port forwarding is used to allow a device from the outside network to access a PC having a reserved DHCP IP address remotely using RDP. It uses a TCP protocol with port number 3389.

The RDP connection coming from any address gets mapped to target IP address 192.168.0.100.

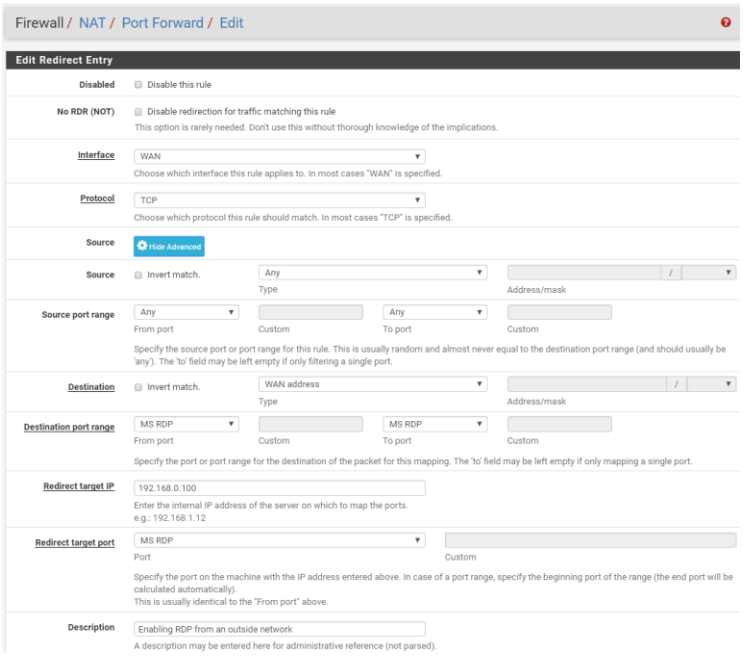


Figure 20 Port forwarding settings

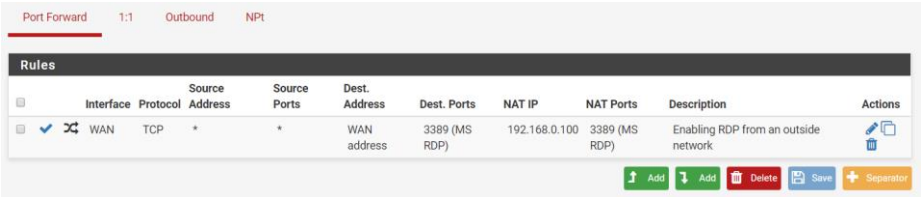


Figure 21 Port forward rule

When a port forwarding rule gets created, it gets added into WAN rules automatically to enable a rule which allows port forwarding. By default, the priority of NAT rules is more than firewall rules.

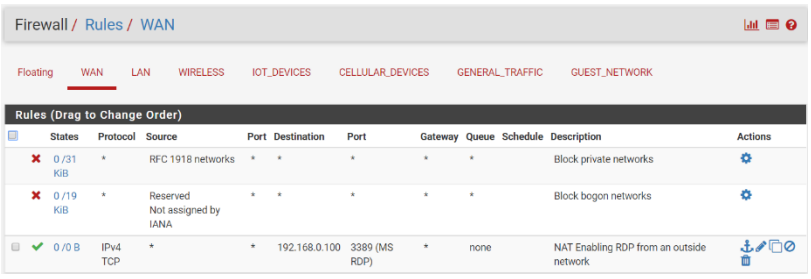


Figure 22 WAN rules

6. Firewall

pfSense is a stateful firewall which features dynamic packet filtering, supports routing based on source IP, destination IP, port type and operating systems. It provides defence against spoofing exploitations by dynamically keeping track of connection information.

Major types of attacks are:

- TCP SYN flooding
- Distributed Denial of Service attack (DDoS)
- DNS Spoofing (Malicious cache poisoning)

6.1. Stateful Firewall

pfSense maintains a stateful firewall in contrary to DSL routers at home. The Internet is an untrusted network of devices, and it makes it very important to sustain stateful firewall which has important features like:

- Security policies
- Rules are stating which packet can pass through the firewall.
- Dynamically keep track of connection information.
- Maintains a state table made up of Source and Destination addresses, port numbers, sequencing and flag information.
- Provides defence against spoofing exploits.

6.2. Firewall Rules

pfSense firewall has many advanced features like Stateful Packet Inspection (SPI), reverse proxy, inbound and outbound NAT mapping, DNS forwarding, anti-spoofing and many. These features make it stand out amongst other firewall and DSL home router.

The firewall rules can be implemented on any interface. Using firewall rules, many functions can be done, like, allowing and disallowing a specific host or network, protocol, OS fingerprinting, aliases and more.

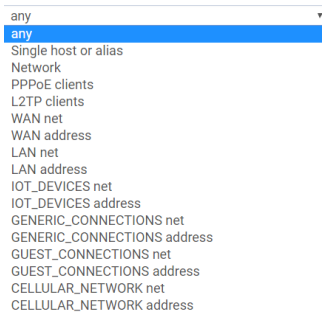


Figure 23 Available networks for setting firewall rules

It is critical to set the action accordingly.

- Pass allows all the traffic to come into the firewall. It is usually used to allow a firewall rule to let certain traffic in and out of the firewall.
- Reject is usually not preferred because it sends a reply back to the program/host telling the 'packet was dropped'. This gives an outside attacker some knowledge that something is there.
- Block drops the packet, and nothing is sent back. Therefore, the attacker cannot know whether or not there is an IP address they are trying to reach.

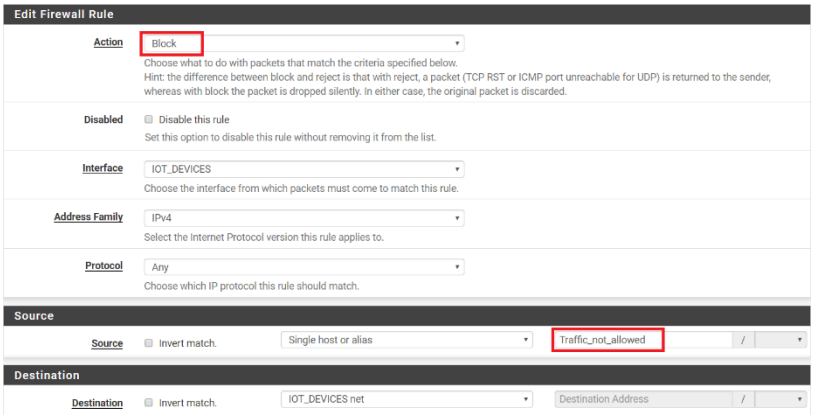


Figure 24 Implementing Block on Alias

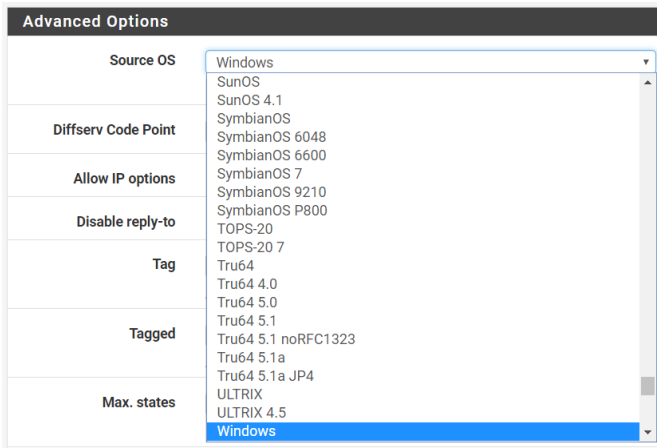


Figure 25 Blocking source on the basis of OS fingerprint

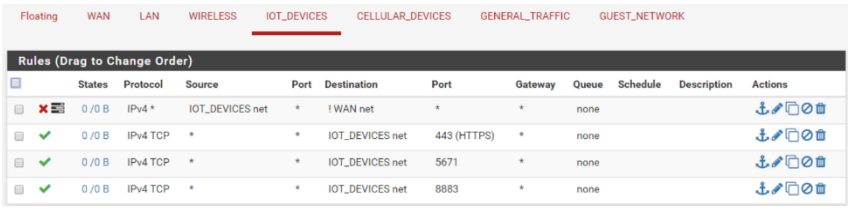


Figure 26 IoT devices network firewall rules

6.1. Firewall Aliases

Alias is a group of networks or hosts IP addresses which are used in firewall rules such that it gets resolved according to the alias list, some changes to be made to a particular host or port can be minimized, and alias name is used in traffic shaper and traffic monitoring. It can incorporate multiple hosts inside a single alias, hence allows fewer firewall rules to be required and saves firewall processing.

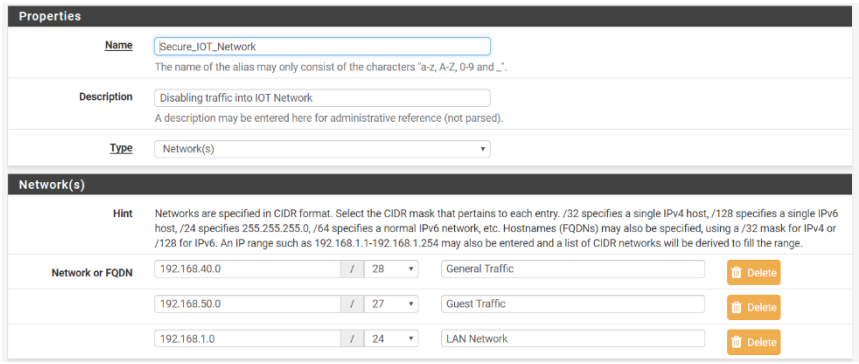


Figure 27 Alias for rejecting traffic into IoT Network

By default, the firewall rules are implemented top to bottom. If there is no rule, no traffic is allowed to pass through that interface. Firewall rules are created so that:

- ‘IoT Devices’ network will block any traffic coming from ‘General Traffic’ and ‘Guest Network’.
- Traffic allowed on ‘IOT Devices network’ is internet only and some specific devices on ‘Cellular Devices’ because some IoT devices (like Chromecast or Google Home) require talking to local devices in order to function entirely by using mDNS protocol.
- Only ports 443 (HTTPS), 5671(AMQP) and 8883(MQTT) are allowed over IOT networks which prevents vulnerability of attacks coming through open ports on the firewall. These are the protocols associated with IoT devices.

6.2. mDNS

mDNS protocol resolves hostnames to IP addresses within a small network. When an mDNS client resolves a hostname, it requests an IP multicast to the host to identify itself.

Avahi protocol maintains the mDNS lookup. This service is only used for certain devices (e.g., Alexa or Chromecast) which need to interact with other devices for functioning.

How mDNS functions with IoT devices are:

- The rules set in this case is, ‘Cellular Network’ has full access to the internet and devices in ‘IOT Network.’
- Any device in ‘Cellular Network’ initiates a request to an IoT device and IOT device will only send back data based on the request. On the other hand, ‘IOT Network’ is unfamiliar about any other network and cannot initiate any request on its own.
- Firewall rules maintain the separation between devices in both the networks.
- Other IOT devices do not need mDNS lookup because they contact with their host server as a service.
- No traffic is being allowed over the two networks; a DNS list is published on both the network sides.

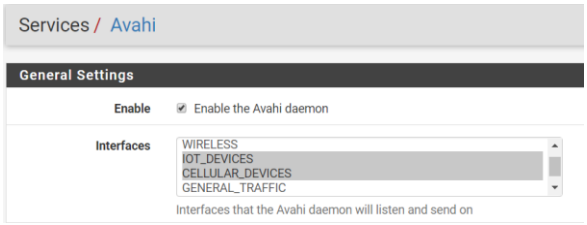


Figure 28 mDNS allowed for selected networks

6.3. pfBlockerNG

pfBlockerNG provides an additional layer of security by blocking external threats and reports malicious incoming connections by logging them. It integrates Pi Hole which prevents DNS requests for known tracking and advertising domains. pfBlockerNG allows the collection of IP address and domain names from a multitude of sources and varying formats and normalises the traffic flowing into the firewall.

This firewall rule precedes every other firewall rule, and it also allows to create aliases that block IP addresses and malicious URLs. Some aliases created are as follows:

- BinaryDefence: It is the collection of lists of known malicious IPs. These lists are from reputable sources and are updated every hour.
- CNCs and BOTnets: It is a collection of exit node lists which can be a distribution method of CNCs and botnets.

- Mail Spammers: It is a collection of mail specific IP list of known email spammers.
- Whitelist List: It is a trusted list of user-defined whitelisted IPs. These are manually added to the custom list. An auto-firewall rule will be created to permit outbound traffic to these IPs.

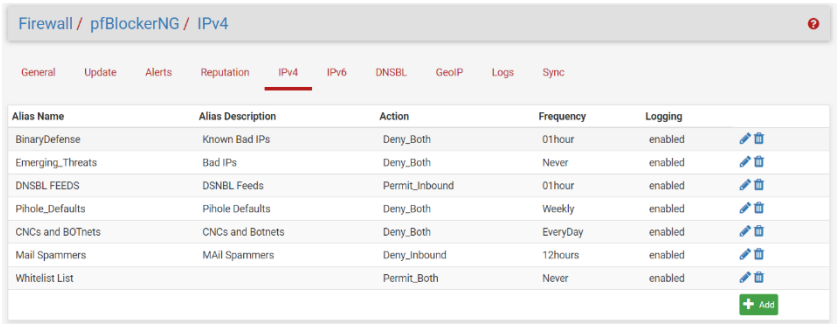


Figure 29 pfBlockerNG Custom Aliases

The rules from pfBlockerNG aliases are imported to LAN and WAN rules.

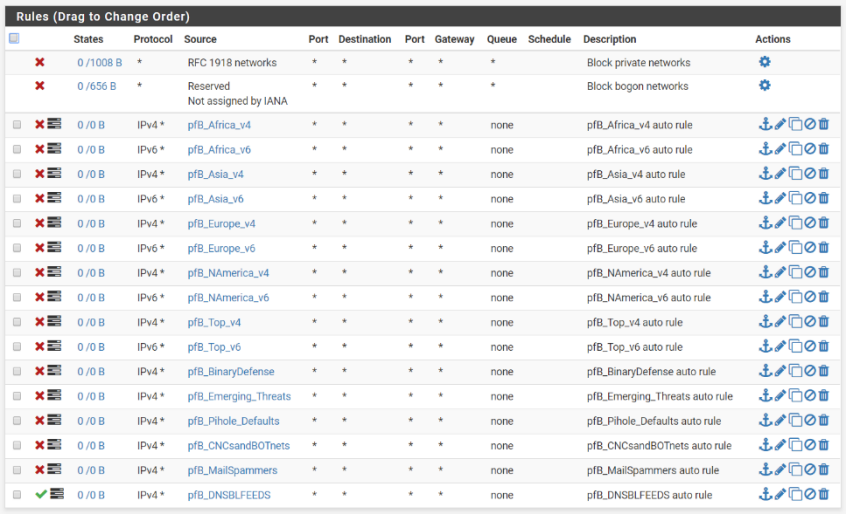


Figure 30 Firewall rules: WAN

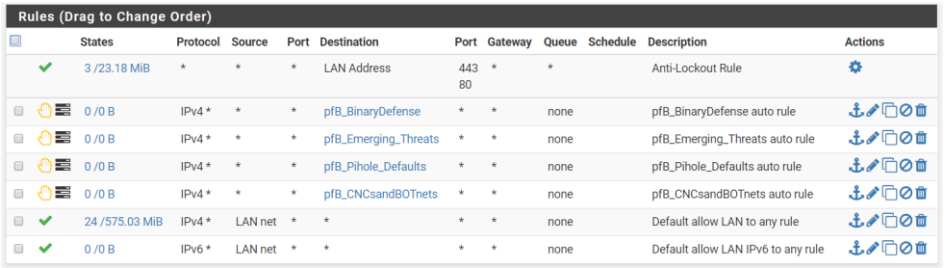


Figure 31Firewall rules: LAN

The results of the implementation of pfBlockerNG, DNSBL and Pi-Hole have filtered the traffic from malicious IP address all over the world. Logs get generated whenever a spam

URL gets filtered in the firewall. The rules are written based on existing knowledge about lists which are used to track malware command and control, spyware, tor nodes and other sorts of malware.

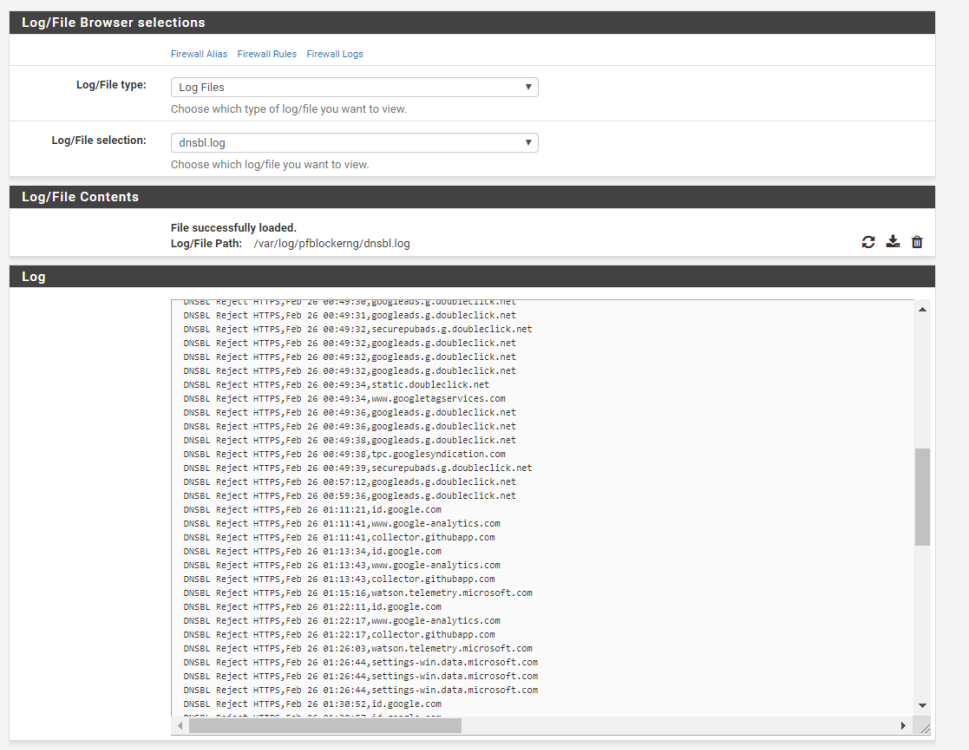


Figure 32 DSNBL Log

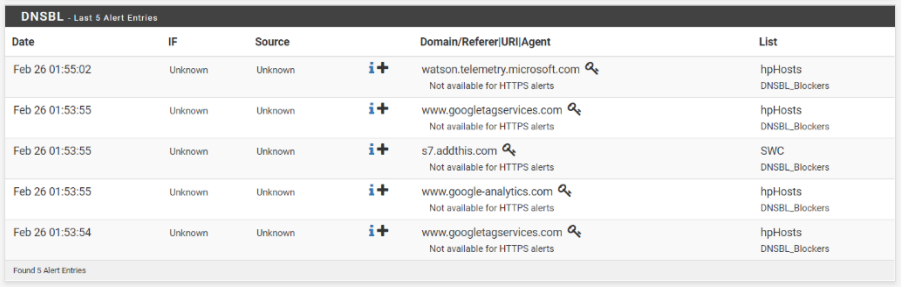


Figure 33 DSBL Alerts

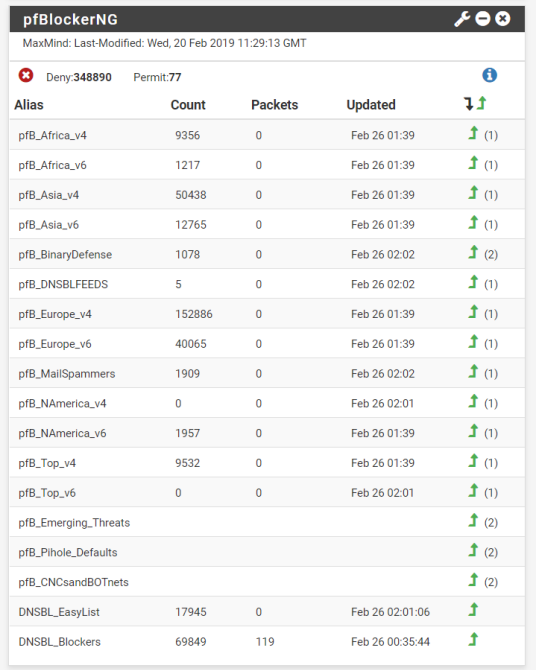


Figure 34 Packet permit and deny status

6.1. Squid

Squid is a web proxy server on the network to which other devices connect to, to connect to the internet. It is important from a security point of view as it sets up an ACL which say where the devices cannot connect. It can get stored in a text file, and it whitelists or blacklists the websites based on the URLs in ACL. Squid enables to force DNS IPV4 lookup first. A transparent mode is enabled which forwards all requests for destination port 80 (HTTP) to the proxy server, filtering out port 443 (HTTPS).

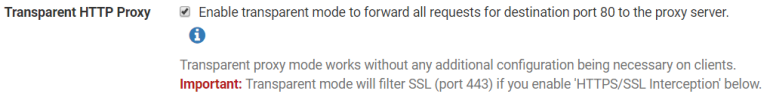


Figure 35 Transparent HTTP proxy

'X-forwarded for header' is a common procedure for identifying the source IP address of a client that connects to a web server through an HTTP proxy. By default, it is enabled and gives out the IP address when a connection to a web server establishes. Delete removes the IP address so that the source host remains anonymous.

It also displays the proxy host's hostname which is 'Capstone_pfsense'.

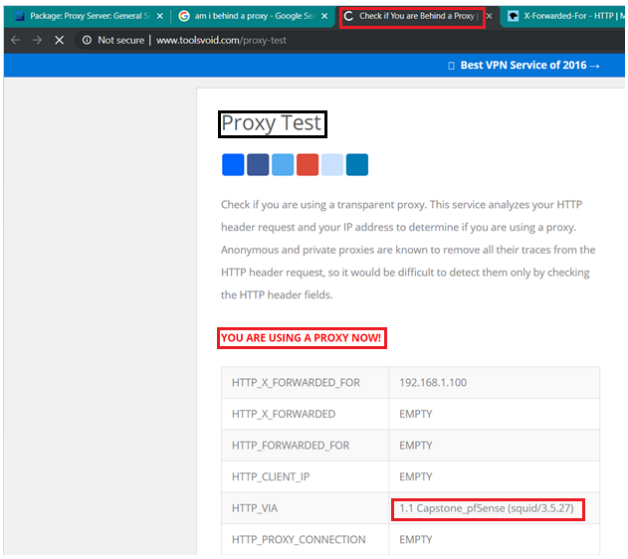


Figure 36 X-Forwarded Header mode ON

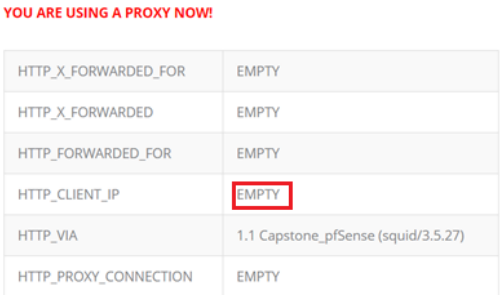


Figure 37 X-Forwarded Header mode delete

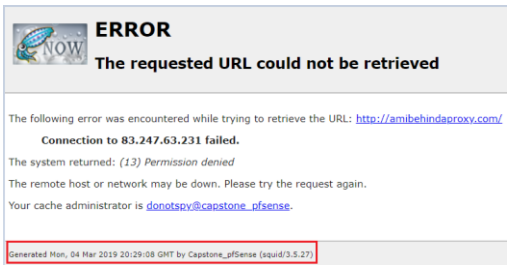
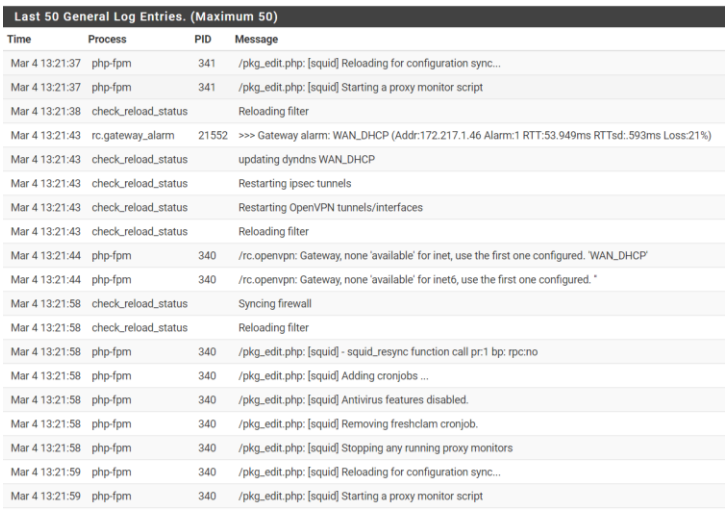


Figure 38 Permission denied for HTTP access

Squid logs log entries in the system firewall status, generating alarms, reloading and starting scripts and also detailing about the squid's antivirus scan reports. It gives out detailed proxy reports about the running process and process ID (PID).



Last 50 General Log Entries. (Maximum 50)

Time	Process	PID	Message
Mar 4 13:21:37	php-fpm	341	/pkg_edit.php: [squid] Reloading for configuration sync...
Mar 4 13:21:37	php-fpm	341	/pkg_edit.php: [squid] Starting a proxy monitor script
Mar 4 13:21:38	check_reload_status		Reloading filter
Mar 4 13:21:43	rc.gateway_alarm	21552	>>> Gateway alarm: WAN_DHCP (Addr:172.217.1.46 Alarm:1 RTT:53.949ms RTTsd:.593ms Loss:21%)
Mar 4 13:21:43	check_reload_status		updating dyndns WAN_DHCP
Mar 4 13:21:43	check_reload_status		Restarting ipsec tunnels
Mar 4 13:21:43	check_reload_status		Restarting OpenVPN tunnels/interfaces
Mar 4 13:21:43	check_reload_status		Reloading filter
Mar 4 13:21:44	php-fpm	340	/rc.openvpn: Gateway, none 'available' for inet, use the first one configured. 'WAN_DHCP'
Mar 4 13:21:44	php-fpm	340	/rc.openvpn: Gateway, none 'available' for inet6, use the first one configured. ''
Mar 4 13:21:58	check_reload_status		Syncing firewall
Mar 4 13:21:58	check_reload_status		Reloading filter
Mar 4 13:21:58	php-fpm	340	/pkg_edit.php: [squid] - squid_resync function call pr:1 bp: rpc:no
Mar 4 13:21:58	php-fpm	340	/pkg_edit.php: [squid] Adding cronjobs ...
Mar 4 13:21:58	php-fpm	340	/pkg_edit.php: [squid] Antivirus features disabled.
Mar 4 13:21:58	php-fpm	340	/pkg_edit.php: [squid] Removing freshclam cronjob.
Mar 4 13:21:58	php-fpm	340	/pkg_edit.php: [squid] Stopping any running proxy monitors
Mar 4 13:21:59	php-fpm	340	/pkg_edit.php: [squid] Reloading for configuration sync...
Mar 4 13:21:59	php-fpm	340	/pkg_edit.php: [squid] Starting a proxy monitor script

Figure 39 Squid Logs

6.1.1. Light Squid

Light Squid is a squid log analyser that parses through the proxy access logs and produces web-based reports detailing the URLs used by each user.

Using port 7445 (default) opens a squid user access report dashboard which is password secured.

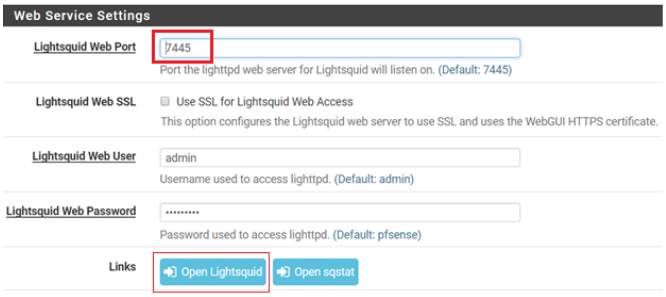


Figure 40 Lightsquid web port

The ‘squid user access report’ dashboard allows viewing the reports of all traffic which have been allowed by through the proxy in an organised list.

Squid user access report
Work Period: Mar 2019

Calendar											
2019											
01	02	03	04	05	06	07	08	09	10	11	12

Date	Group	Users	Oversize	Bytes	Average	Hit %
06 Mar 2019	grp	2	0	55 517	27 758	0.00%
05 Mar 2019	grp	1	0	35 755	35 755	0.00%
04 Mar 2019	grp	1	0	9.1 M	9.1 M	0.60%
Total/Average:		1	0	9.2 M	3.1 M	0.20%

LightSquid v1.8 (c) Sergey Erokhin AKA ESL

Figure 41 Squid user access report dashboard

Total					
			9.1 M		
#	Accessed site	Connect	Bytes	Cumulative	%
1	img-s-msn-com.akamaized.net	7	4.5 M	4.5 M	49.6%
2	www.shallalist.de	14	3.9 M	8.5 M	43.0%
3	qurl.cloud.360safe.com	90	218 049	8.7 M	2.2%
4	a.optnmstr.com	1	194 027	8.8 M	2.0%
5	www.toolsvoid.com	13	53 538	8.9 M	0.5%
6	http://capstone_pfsense:3128/squid-internal-static/icons/SN.png	4	52 348	8.9 M	0.5%
7	whatismyip.network	11	42 992	9.0 M	0.4%
8	www.whatismyproxy.com	8	35 515	9.0 M	0.3%
9	content.dellsupportcenter.com	8	27 244	9.0 M	0.2%
10	amibehindaproxy.com	6	26 054	9.1 M	0.2%
11	cdn.content.prod.cms.msn.com	4	12 181	9.1 M	0.1%
12	ocsp.digicert.com	11	10 164	9.1 M	0.1%
13	lagado.com	2	8 584	9.1 M	0.0%
14	iup.360safe.com	1	8 143	9.1 M	0.0%
15	ccleaner.tools.avcdn.net	2	2 608	9.1 M	0.0%
16	sdup.update.360safe.com	1	1 598	9.1 M	0.0%
17	tconf.cloud.360safe.com	1	974	9.1 M	0.0%
18	ip-info.ff.avast.com	1	642	9.1 M	0.0%
19	www.ebay.com	1	625	9.1 M	0.0%
20	104.192.108.113	1	548	9.1 M	0.0%
21	emupdate.avcdn.net	1	356	9.1 M	0.0%
Total			9.1 M		

Figure 42 Accessed URLs log

6.1.2. SquidGuard

SquidGuard is a URL redirector software which is used to allow or deny access to specific URLs. It gives the flexibility to modify general categories which can then be allowed, whitelist or blacklist.

- Allow – Lets the URL pass if it is not denied in other categories.
- Whitelist – Lets the URL to be always passed.
- Blacklist – Does not allow the URL to be accessed.

Target categories are group ACLs which have target rules list. It is categorized according to the user-specific domain whitelisting or blocking. Each value in the row of target category has allowed, deny or whitelist.

Name	Redirect	Description	
Blacklist_4_IOT			
Whitelist_4_IOT			
Blacklist			
			Add

Figure 43 Custom target categories

Whitelist domain list contains the URLs which are secure, and the IoT devices will need to get updates from and visit frequently.

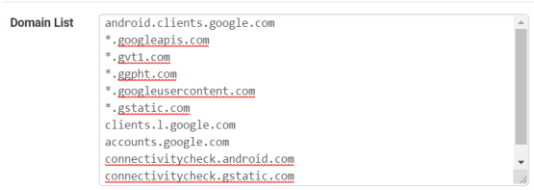


Figure 44 Whitelist for IoT devices network

Apart from custom target categories, there are several inbuilt categories which are listed and are allowed or denied according to the category they belong. As an example, blk_BL_drugs is a category which is denied.

Shallalist is a collection of URL lists grouped into several categories having reliable information of malicious websites and domains.

Figure 45 URL blacklist: Shallalist

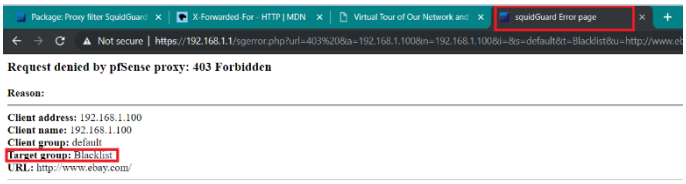


Figure 46 Test blacklist output

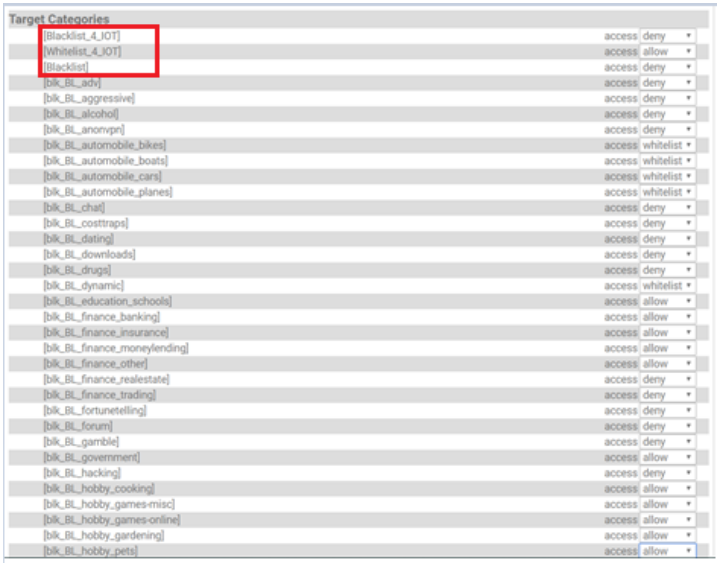


Figure 47 ACLs with the whitelist, deny, allow

7. IPS/IDS

Intrusion Detection System is a network security tool which identifies, assess and reports unauthorised or unapproved network activity. An IDS detect and deal with insider attacks and external attacks. Network-based IDS uses packet sniffing techniques to pull data from TCP/IP packets and other protocols which are travelling along the network.

An IDS works by scanning for a known identity or a signature for each specific intrusion event. It does so by regularly receiving signature updates stored in a reliable database. It also monitors, logs and reports detected malicious activities.

Intrusion Prevention System is a network tool that is configured to block potential threats. It creates a security barrier behind the firewall and provides a complimentary layer of analysis that IDS detects. The functions of IPS are:

- Sending an alarm to the administrator when malicious packets get identified
- Dropping malicious packets
- Blocking traffic from the source address
- Resetting the connection

Snort is an open source IDS/IPS that makes identification and blocking by scanning predefined lists. It works by downloading definitions that it uses to inspect packets passing through the firewall. It then generates logs and alerts based on suspicious traffic.

7.1. Configuring Snort

Snort is an open source IDS/IPS that perfroms real-time traffic analysis and packet logging on IP networks. It is capable of performing protocol analysis, content searching, and matching, and is used to detect attacks such as stealth port scans, CGI attacks, and SMB probes.

A snort oinkmaster code is required to initiate the snort configuration. This code is a unique key associated with the user account and acts as an API key for downloading rules from trusted URLs.

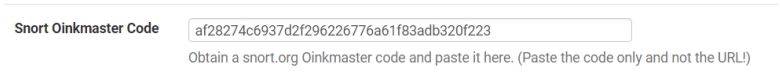
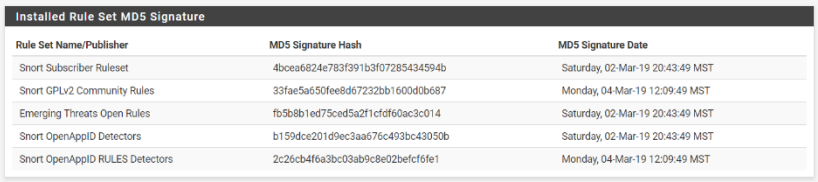


Figure 48 Oinkmaster Code

The intelligence of snort comes from having the rules to identify and apply the information to the traffic passing through the interface in order to detect and act upon the detected malicious traffic. Snort rules are designed such that it describes the following events accurately:

- The condition in which a user thinks that a network packet's identity is not authentic.
- Any violation of the security policies that might be a threat to the network and reveal valuable information.

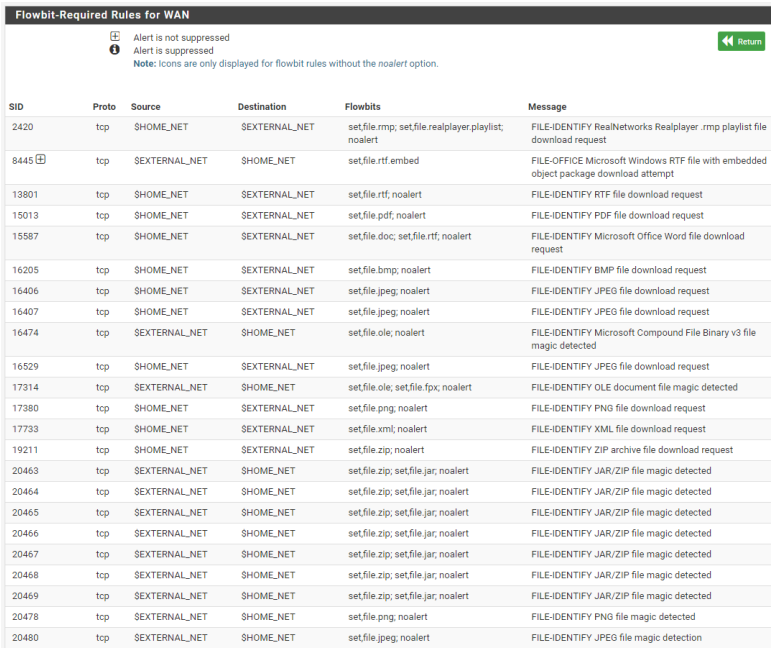


Installed Rule Set MD5 Signature

Rule Set Name/Publisher	MD5 Signature Hash	MD5 Signature Date
Snort Subscriber Ruleset	4bcea6824e783f391b3f07285434594b	Saturday, 02-Mar-19 20:43:49 MST
Snort GPLv2 Community Rules	33fae5a650fee8d67232bb1600d0b687	Monday, 04-Mar-19 12:09:49 MST
Emerging Threats Open Rules	fb5b8b1ed75ced5a2f1cfdf60ac3c014	Saturday, 02-Mar-19 20:43:49 MST
Snort OpenAppID Detectors	b159dce201d9ec3aa676c493bc43050b	Saturday, 02-Mar-19 20:43:49 MST
Snort OpenAppID RULES Detectors	2c26cb4f6a3bc03ab9c8e02befcf6fe1	Monday, 04-Mar-19 12:09:49 MST

Figure 49 Snort rule set

Snort uses flowbits detection plugins that use flow preprocessor to track rule state during a transport protocol session. It allows rules to track the state of an application protocol generically.



Flowbit-Required Rules for WAN

Alert is not suppressed
Alert is suppressed
Note: Icons are only displayed for flowbit rules without the *noalert* option.

Return

SID	Proto	Source	Destination	Flowbits	Message
2420	tcp	$HOME_NET	$EXTERNAL_NET	set,file.rmp; set,file.realplayer.playlist; noalert	FILE-IDENTIFY RealNetworks Realplayer .rmp playlist file download request
8445	tcp	$EXTERNAL_NET	$HOME_NET	set,file.rtf.embed	FILE-OFFICE Microsoft Windows RTF file with embedded object package download attempt
13801	tcp	$HOME_NET	$EXTERNAL_NET	set,file.rtf; noalert	FILE-IDENTIFY RTF file download request
15013	tcp	$HOME_NET	$EXTERNAL_NET	set,file.pdf; noalert	FILE-IDENTIFY PDF file download request
15587	tcp	$HOME_NET	$EXTERNAL_NET	set,file.doc; set,file.rtf; noalert	FILE-IDENTIFY Microsoft Office Word file download request
16205	tcp	$HOME_NET	$EXTERNAL_NET	set,file.bmp; noalert	FILE-IDENTIFY BMP file download request
16406	tcp	$HOME_NET	$EXTERNAL_NET	set,file.jpeg; noalert	FILE-IDENTIFY JPEG file download request
16407	tcp	$HOME_NET	$EXTERNAL_NET	set,file.jpeg; noalert	FILE-IDENTIFY JPEG file download request
16474	tcp	$EXTERNAL_NET	$HOME_NET	set,file.ole; noalert	FILE-IDENTIFY Microsoft Compound File Binary v3 file magic detected
16529	tcp	$HOME_NET	$EXTERNAL_NET	set,file.jpeg; noalert	FILE-IDENTIFY JPEG file download request
17314	tcp	$EXTERNAL_NET	$HOME_NET	set,file.ole; set,file.fpx; noalert	FILE-IDENTIFY OLE document file magic detected
17380	tcp	$HOME_NET	$EXTERNAL_NET	set,file.png; noalert	FILE-IDENTIFY PNG file download request
17733	tcp	$HOME_NET	$EXTERNAL_NET	set,file.xml; noalert	FILE-IDENTIFY XML file download request
19211	tcp	$HOME_NET	$EXTERNAL_NET	set,file.zip; noalert	FILE-IDENTIFY ZIP archive file download request
20463	tcp	$EXTERNAL_NET	$HOME_NET	set,file.zip; set,file.jar; noalert	FILE-IDENTIFY JAR/ZIP file magic detected
20464	tcp	$EXTERNAL_NET	$HOME_NET	set,file.zip; set,file.jar; noalert	FILE-IDENTIFY JAR/ZIP file magic detected
20465	tcp	$EXTERNAL_NET	$HOME_NET	set,file.zip; set,file.jar; noalert	FILE-IDENTIFY JAR/ZIP file magic detected
20466	tcp	$EXTERNAL_NET	$HOME_NET	set,file.zip; set,file.jar; noalert	FILE-IDENTIFY JAR/ZIP file magic detected
20467	tcp	$EXTERNAL_NET	$HOME_NET	set,file.zip; set,file.jar; noalert	FILE-IDENTIFY JAR/ZIP file magic detected
20468	tcp	$EXTERNAL_NET	$HOME_NET	set,file.zip; set,file.jar; noalert	FILE-IDENTIFY JAR/ZIP file magic detected
20469	tcp	$EXTERNAL_NET	$HOME_NET	set,file.zip; set,file.jar; noalert	FILE-IDENTIFY JAR/ZIP file magic detected
20478	tcp	$EXTERNAL_NET	$HOME_NET	set,file.png; noalert	FILE-IDENTIFY PNG file magic detected
20480	tcp	$EXTERNAL_NET	$HOME_NET	set,file.jpeg; noalert	FILE-IDENTIFY JPEG file magic detection

Figure 50 Flowbit required rules

Snort uses predefined IPS policies to prevent the malicious detected packets from entering into the network. Policy selection 'Security' enables most of the rules and results in higher false positives.

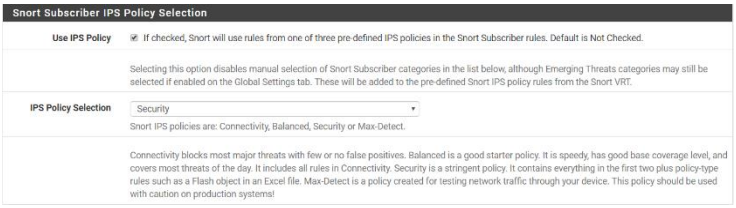


Figure 51 IPS policy Security

The members of Snort Integrators submit snort community rules. These rulesets are developed with intensive analysis of packet captures of the data.

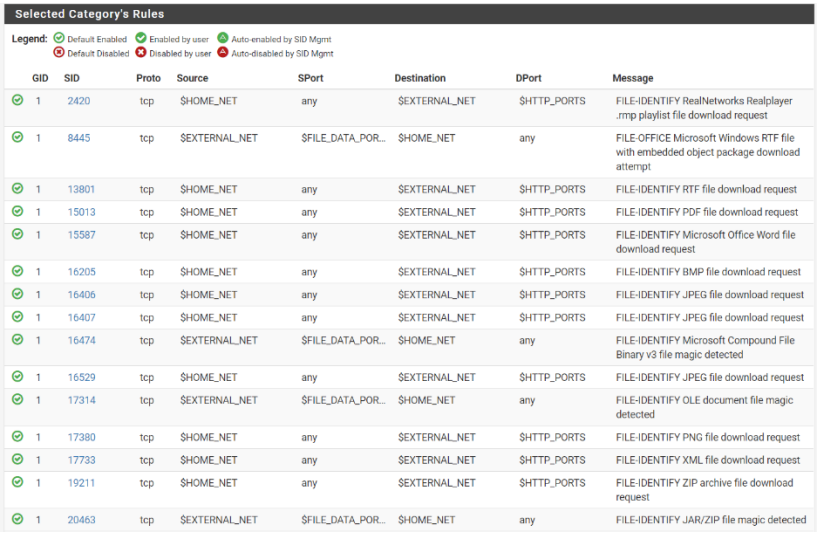


Figure 52 Snort Community Rules

Snort IDS detects all kinds of alerts by seeing activities that pass through the firewall. It identifies parameters like source and destination ports, IP and describes the kind of alert.

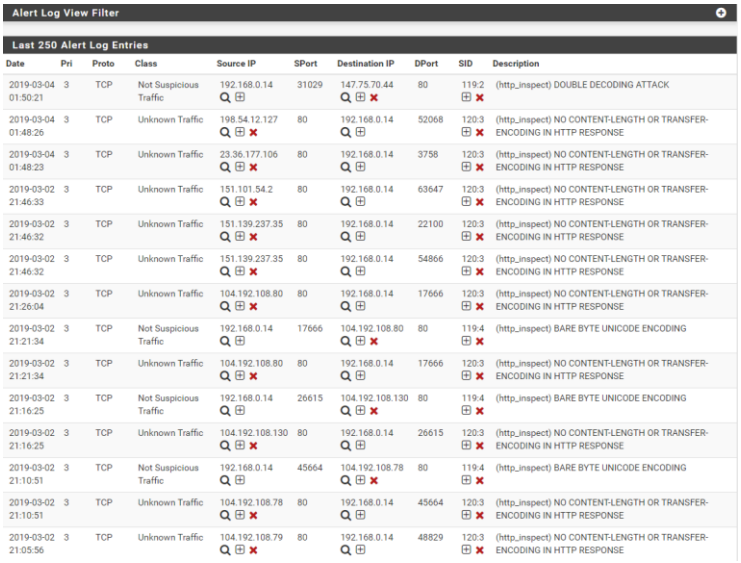


Figure 53 Snort alert log

Snort IDS detected an alert named Double decoding attack and IPS prevented it from passing through the firewall. This event is caused when double encoded characters are detected in web traffic. Such unusual behaviour indicates a possible attack against a vulnerable system. A possible scenario is that an attacker might double encode the request to the web server.

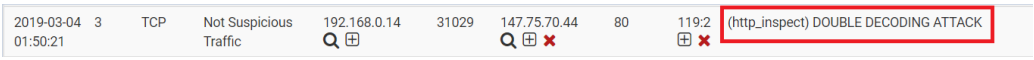


Figure 54 Double decoding attack

8. UUID

UUID is a 128 bits number used to identify a device connected over the internet. UUID guarantees uniqueness as it is a combination of the IP address or MAC address which is hard-wired on NIC of the host and current timestamp.

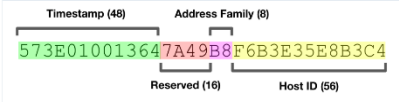


Figure 55 128-bit UUID format

UUID has five versions:

- Version 1: Time based UUID, the timestamp is a 60-bit value with a combination of MAC address. It is represented by UTC (Coordinated Universal Time) as a count of 100 nanosecond interval. For the systems which do not have UTC, the local time is used.
- Version 2 is a date, time and MAC address with DCE security version.
- Version 3 and 5 are created by hashing and namespace identifier that uses MD5 and SHA1 respectively.
- Version 4 is a randomly generated UUID.

8.1. UUID Version 1

To ensure the uniqueness of the identity of a device which connects wirelessly, and use it as a connection parameter, UUID has to be the same every time (except current date and time); hence version 1 is used. Version 1 UUID is generated by the computer's MAC address and current date and time. Therefore, a completely uniquely ID is generated every time such that it doesn't collide with any other device.

A python program for generating IP and MAC address and UUID is programmed such that it finds the above parameters and displays them.

pfSense has proposed to change most of its back-end language from php to python and using this program with compatible pfSense libraries; this program can autorun and save these parameters to its database when a new connection is made. With this database, the device can be recognized as a new or existing device connecting to the pfSense firewall.

UUID is unique to every device and adds another layer of security, and 128-bit numbers make it impossible to guess. Without the physical access to a device, UUID cannot be spoofed which makes it a reliable connection parameter.

8.2. Program

```
import socket
import re, uuid

def host_IP():
  try:
    hname = socket.gethostname()
    hip = socket.gethostbyname(hname)
    print("Hostname:  ",hname)
    print("IP Address: ",hip)
  except:
    print("Unable to get Hostname and IP")

host_IP()
print ("MAC address: ", end="")
print (':'.join(re.findall('..', '%012x' % uuid.getnode())))

#Printing UUID and related parameters
UUID = uuid.uuid1()
print ("UUID: ", (UUID))
print("UUID Type:  ",type(UUID))
#print('UUID.bytes   :', UUID.bytes)
#print('UUID.bytes_le :', UUID.bytes_le)
#print('UUID.hex     :', UUID.hex)
#print('UUID.int     :', UUID.int)
#print('UUID.urn     :', UUID.urn)
#print('UUID.variant :', UUID.variant)
#print('UUID.version :', UUID.version)
#print('UUID.fields  :', UUID.fields)
```

```
#print('UUID.time_low           : ', UUID.time_low)
#print('UUID.time_mid           : ', UUID.time_mid)
#print('UUID.time_hi_version    : ', UUID.time_hi_version)
#print('UUID.clock_seq_hi_variant: ', UUID.clock_seq_hi_variant)
#print('UUID.clock_seq_low      : ', UUID.clock_seq_low)
#print('UUID.node               : ', UUID.node)
#print('UUID.time               : ', UUID.time)
#print('UUID.clock_seq          : ', UUID.clock_seq)
#print('UUID.SafeUUID           : ', UUID.is_safe)

key=input("Press any key to continue   :   ")
```

The python version 3.5 is used to write and compile the program in IDE 3.5. Socket, re and uuid are inbuilt python libraries which are called in the program; where socket establishes a socket connection, re extracts the information from the device in a human-friendly UUID format and UUID extracts the UUID information from the device based on which version is used.

The IP address is fetched from the system with the help of system hostname. Try and exception are used in case IP address and hostname are not fetched due to some reason.

Re.findall matches all the UUID parameters and displays the information in reference to the system's signature. All the commented outputs are UUID parameters which are matched by UUID library and only the most relevant are displayed.

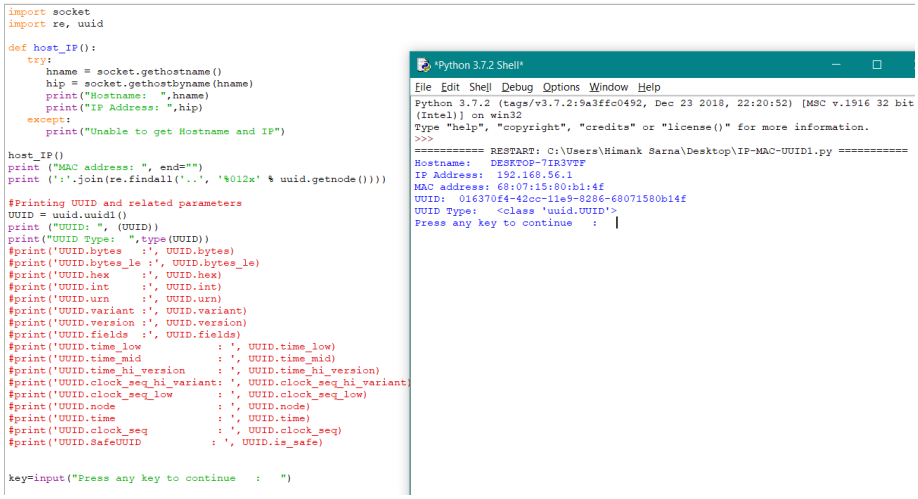


Figure 56 Output in IDE

Pip is a python tool for installing and managing packages. With the help of autopytoexe package, pip converts a .py file to .exe.

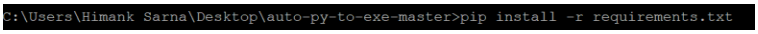


Figure 57 Installing pip install for autopytoexe

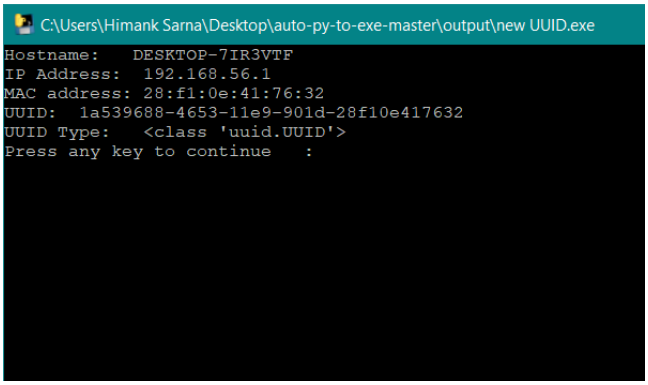


Figure 58 Output in .exe

This program prints the IP address, MAC address and version 1 UUID of the device this program runs on. UUID library in python which enables functions uuid1(), uuid3(), uuid4(), uuid5() for generating respective UUID versions as specified in RFC 4122.

Field	Meaning
time_low	the first 32 bits of the UUID
time_mid	the next 16 bits of the UUID
time_hi_version	the next 16 bits of the UUID
clock_seq_hi_variant	the next 8 bits of the UUID
clock_seq_low	the next 8 bits of the UUID
node	the last 48 bits of the UUID
time	the 60-bit timestamp
clock_seq	the 14-bit sequence number

Figure 59 UUID fields

Executing this program as an autorun executable in pfSense, it can save the UUID of a new device connecting to it in its database which will be mapped with a specific MAC address. When a different device with the same MAC address but a different UUID attempts to establish a connection with pfSense, the connection will be denied because the new identifiers will not match the existing saved parameters.

9. Conclusion

With the increasing popularity of IoT devices, securing the sensitive data is critical, and pfSense firewall router provides a complete security solution. pfSense is a highly configurable, full-featured solution which addresses security vulnerabilities of IoT and other devices in a network.

It provides great security by segmenting the network and isolating subnetworks, denying the traffic from one VLAN to another.

It restricts unknown hosts to connect to the network and limits the number of hosts connecting through DHCP by limiting the network's subnet mask.

pfSense firewall rules prevent malicious packets from entering the packets and generates alerts for the administrator. It provides control-based access on the basis of source and destination and blocks denied users from entering into the internal network. pfSense also secures the devices by blacklisting untrusted URLs.

Stateful NAT in pfSense rewrites source port on all outgoing packets which reduces IP spoofing. It also blocks all the open ports which lead to security vulnerabilities.

IDS/IPS continuously gather information about the network by identifying potential threats, logging information about them and deterring unauthorized packets to enter into the network.

Integrating UUID with IP and MAC address as a new connection parameter further improves the security of new devices that connect to the network.

Overall, pfSense is a highly reliable firewall router which secures the network and sensitive data from getting compromised. It is highly flexible, and new addon/packages are added with new features oftentimes, which makes it future-oriented and scalable.

10. Appendix A: List of Figures

11. Appendix B: List of Acronyms and Abbreviations

IP	Internet Protocol
CGI	Common Gateway Interface
IPS/IDS	Intrusion Prevention System/ Intrusion Detection System
DHCP	Dynamic Host Configuration Protocol
ARP	Address Resolution Protocol
TCP/IP	Transmission Control Protocol/ Internet Protocol
DNS	Domain Name System
OS	Operating System
SMB	Server Message Block
URL	Uniform Resource Locator
API	Application Program Interface
ACL	Access Control List
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
GUI	Graphical User Interface
PID	Process Identifier
SSID	Service Set Identifier
DCE	Data Communications Equipment
AP	Access Point
UUID	Universally Unique Identifier
MAC	Media Access Control
SPI	Stateful Packet Inspection
IOT	Internet of Things
OPT	Optional
PMK	Pairwise Master Key
UPnP	Universal Plug and Play

12. Appendix C: References

- https://en.wikipedia.org/wiki/Universally_unique_identifier
- https://tools.ietf.org/html/rfc4122
- https://stackoverflow.com/questions/292965/what-is-a-uuid
- https://docs.python.org/3/library/uuid.html
- https://www.netgate.com/solutions/pfsense/features.html
- https://www.youtube.com/watch?v=BwK2cal4SOQ
- https://www.reddit.com/r/PFSENSE/comments/7c2vrm/pfblockerng_ip_lists_dnsbl_feed
- https://forum.netgate.com/topic/102290/lists-for-pfblockernghttps://www.tecmint.com/install-configure-pfblockerng-dns-black-listing-in-pfsense/
- https://www.tecmint.com/install-configure-pfblockerng-dns-black-listing-in-pfsense/
- https://searchsecurity.techtarget.com/definition/intrusion-detection-system
- https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips
- https://wpollock.com/AUnixSec/NIDS-snort.htm
- https://cqureacademy.com/blog/penetration-testing/smb-relay-attack
- https://networkdirection.net/articles/asa/firepowermanagementcentre/firepowerintrusiondetection/
- https://www.snort.org/faq/what-are-community-rules
- https://www.aldeid.com/wiki/Snort-alerts/http_inspect-DOUBLE-DECODING-ATTACK
- https://www.snort.org/faq/readme-flowbits
- https://en.wikipedia.org/wiki/X-Forwarded-For
- https://turbofuture.com/computers/Introduction-to-pfSense-An-Open-Source-Firewall-and-Router-Platform
- https://github.com/pfsense/pfsense/find/v2.4.4_1
- https://www.youtube.com/watch?v=-FFGJGPzZJw&list=PLcuqiGkigdex_2ytK5X-HptOJSJ7Mm9bU

- https://turbofuture.com/internet/How-to-Set-Up-an-Intrusion-Detection-System-Using-Snort-on-pfSense-20
- https://community.spiceworks.com/how_to/126207-set-up-snort-on-pfsense-for-ids-ips
- https://docs.netgate.com/pfsense/en/latest/dhcp/dhcp-search-domains-on-windows-dhcp-clients.html
- https://en.wikipedia.org/wiki/List_of_HTTP_header_fields
- https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips
- https://www.snort.org/faq/what-is-snort