Author: mow9902 of the Puppy Linux Discussion Forum

FW Buddy (aka FWBuddy)

User Manual

"An Application to assist non technical users to manage their iptables Firewall"

Table of Contents

Disclaimer-1:

You use this software at your own risk. It is your responsibility to ensure that the software works on your system with no negative impact on any other system components.

The author makes no guarantees about the compatibility of this software with your system, and accepts no responsibility for negative impact which may occur on your system, or your data.

Version History:

Version	Created	Comments
1.00	23rd Nov 2022	Original version

Background, Purpose, Function

(The following paragraph describes the puppy firewall to the best of my knowledge. I do not claim expert, or complete knowledge of this functionality, and I apologise in advance for any errors or misleading statements.)

The default firewall function on puppy systems is provided by iptables.

When the system is first started, the script ***/etc*/init.d/rc.firewall** writes and loads a default ruleset into iptables, and the script ***/usr.bin/firewallstatus.sh*** places a firewall icon in the system tray to indicate that the firewall is now “Active”

The firewall is “managed” via the inbuilt GUI **/usr/sbin/firewall_ng**

This GUI enables a default ruleset to be generated and loaded into iptables, and also provides some ability to “customize” the firewall by generating and loading specific additional rules to meet your particular needs.
When the system is first started, the script ***/etc*/init.d/rc.firewall** writes and loads a ruleset into iptables reflecting the choices selected using the GU, and places a firewall icon in the system tray to indicate that the firewall is now “Active”

If the user “stops” the firewall (by clicking the option in the firewall icon in the tray), the existing ruleset is totally ‘flushed’ out of iptables, and the tray icon is updated to indicate that the firewall is now “OFF”

If the user “restarts” the firewall (by clicking the option in the firewall icon in the tray), the ruleset is once again written by the script ***/etc*/init.d/rc.firewall** and loaded back into iptables, and the tray icon is updated to indicate that the firewall is now “Active”

Purpose

All aspects of the puppy firewall (iptables) can be managed with either:
- the *inbuilt GU*I (/usr/sbin/firewall_ng) – *for quick changes and/or for those who are “less technical”*
- direct *input of commands via a terminal session* to the iptables program – *for any changes for those who are capable and confident with the complexity of the iptables program.*

FWBuddy (this Application) does not provide any firewall functionality in addition to that which is already provided by the inbuilt tools.
If you are fully comfortable understanding and inputting terminal commands to manage your iptables, then you will not find any use for this application.

I have created this application to address my personal situation in this regard – ie:
I am a “less technical” user and not fully capable or confident with direct input of commands to iptables.
Although the default puppy ruleset provides most of my required firewall functionality for most of the time, there are occasions when I want to make either a permanent or temporary change while I am working with a particular app or on a particular project.

On these occasions I find that I need to review and try to “relearn” how iptables work, and the process and format required to input valid commands. This is time consuming and frustrating.
In this situation I want to:
- Easily understand what the current ruleset is doing
- Easily understand how I might make a small change if needed
- Save the current ruleset and, if necessary, reload it after I have finished whatever changes I

made OR reload it if I manage to make an error
- Load a custom ruleset as/when required
- If I have made changes to the ruleset which I want to apply permanently, I want these custom rules to be loaded whenever the system is restarted.

FWBuddy was created to meet this need. It provides the following functionality:

- Check the current status of the firewall
- Stop the firewall OFF temporarily (can also be done with inbuilt firewall GUI)
- Restart the firewall (can also be done with inbuilt firewall GUI)
- show live statistics of data packets being filtered by the current iptables ruleset
- List in readable format the current iptables ruleset being applied
- Delete a rule from the filter table
- Add/Insert a new rule to the filter table – and also
- Include a comment to indicate the purpose of the rule
- Save the current ruleset to a location of your choice
- Load a custom ruleset to temporarily (or permanently replace the current ruleset)
- Specify a custom ruleset to be loaded whenever the system is started
- Allow for the input of a custom command

Dependencies:

This application uses:
- gtkdialog to present GUIs which are used to issue the underlying system commands
- gtkdialog-splash to present user messages
- yad-FWB (copy of yad) to present user messages

Installation:

This application is fully portable and does not require ‘installation’. Simply extract the downloaded file to a directory of your choice and run it by clicking on the script FWBuddy.sh
Note: If you choose to run the App from a terminal window by executing the same script, then the App will naturally close when/if you close the terminal window.

IPTABLES – Structure/Terminology

An Iptables FW comprises the following: ***(as best I understand it personally!)***

- **Tables** is the name for a set of chains.
  - **Chain** is a collection of rules.
    - **Rule** is condition used to match packet.
      - Target is action taken when a possible rule matches. Examples of the target are ACCEPT, DROP, QUEUE.
      - Policy is the default action taken in case of no match with the inbuilt chains and can be ACCEPT or DROP.

There are five possible **tables**: which can be supplemented and/or with others either by the specific linux kernel in use, the linux distribution, or personally added tables.

- **filter**: **Default used table for packet filtering**.
  - It includes chains like INPUT, OUTPUT and FORWARD.
- **Nat**: Related to Network Address Translation.
  - It includes PREROUTING and POSTROUTING chains.
- **Mangle**: For specialised packet alteration.
  - Inbuilt chains include PREROUTING and OUTPUT.
- **Raw**: Configures exemptions from connection tracking.
  - Built-in chains are PREROUTING and OUTPUT.
- **Security**: Used for **Mandatory Access Control**

The following Tables exist on my Puppy system:

- **Filter**
- Nat
- Mangle

Note: FWBuddy applies rule changes only for the Filter table. At the moment I have no use for the other tables.

There are several default **Chains** which can be supplemented and/or with others either by the specific linux kernel in use, the linux distribution, or personally added chains.

- **INPUT:** set of rules for packets destined to localhost sockets.
- **FORWARD:** for packets routed through the device.
- **OUTPUT:** for locally generated packets, meant to be transmitted outside.
- **PREROUTING:** for modifying packets as they arrive.
- **POSTROUTING:** for modifying packets as they are leaving.

Note: User-defined chains can also be created.

The following Chains exist on my Puppy Linux system

Primary Chains:

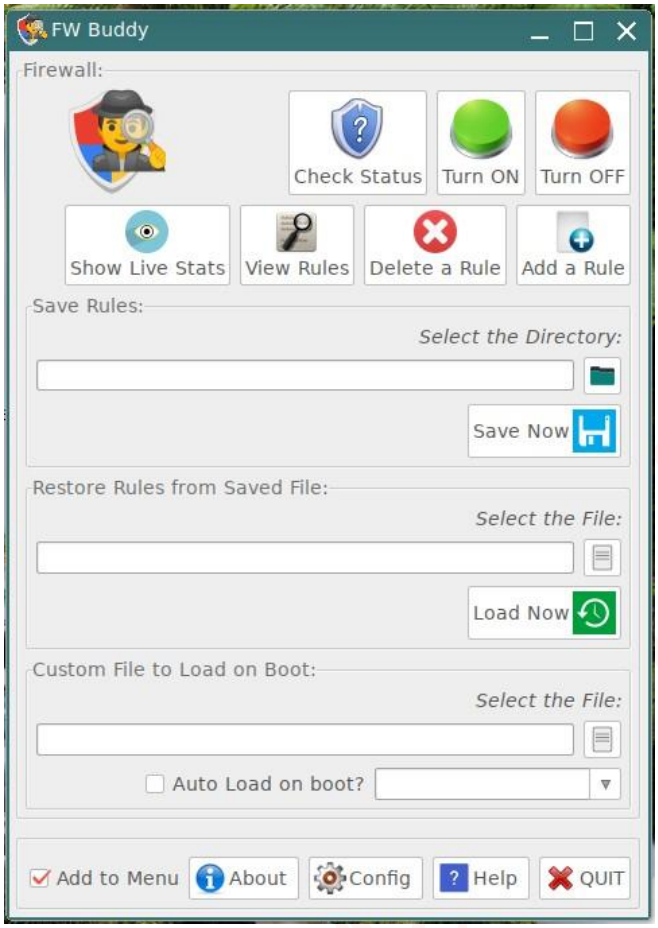
- INPUT
- FORWARD
- OUTPUT

+ **Secondary Chains** called by rules inserted into Primary Chains

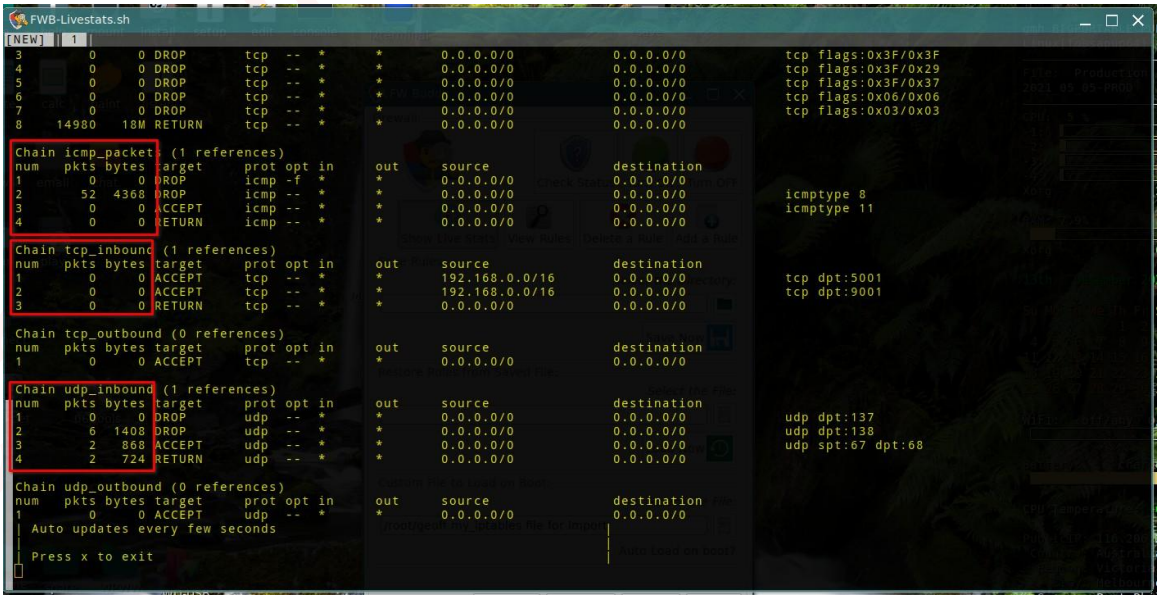
- bad_packets
- bad_tcp_packets
- icmp_packets
- tcp_inbound
- tcp_outbound
- udp_inbound
- udp_outbound

Usage:

Main Screen



- **Check Status** – will show if the firewall is currently ON or OFF
- **Turn ON** – turn it ON
- **Turn OFF** – turn it OFF
- **Show Live Stats**
    - Displays the following screen which shows each of the current rules, and the amount of data which is being processed by the rule.



- **View Rules**
  - Outputs a formatted list of rules for your review.

iptables.txt

```
Chain INPUT (policy DROP)
num  target     prot opt source               destination
1    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
2    bad_packets  all  --  0.0.0.0/0            0.0.0.0/0
3    DROP       all  --  0.0.0.0/0            224.0.0.1
4    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           ctstate RELATED,ESTABLISHED
5    tcp_inbound  tcp  --  0.0.0.0/0            0.0.0.0/0
6    udp_inbound  udp  --  0.0.0.0/0            0.0.0.0/0
7    icmp_packets  icmp --  0.0.0.0/0            0.0.0.0/0
8    DROP       all  --  0.0.0.0/0            0.0.0.0/0           PKTTYPE = broadcast

Chain FORWARD (policy DROP)
num  target     prot opt source               destination

Chain OUTPUT (policy DROP)
num  target     prot opt source               destination
1    DROP       tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:53
2    DROP       udp  --  0.0.0.0/0            0.0.0.0/0           udp dpt:53
3    DROP       icmp --  0.0.0.0/0            0.0.0.0/0           ctstate INVALID
4    ACCEPT     all  --  127.0.0.1            0.0.0.0/0
5    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
6    ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
7    ACCEPT     all  --  0.0.0.0/0            127.0.0.1
8    ACCEPT     all  --  0.0.0.0/0            116.206.228.179

Chain bad_packets (1 references)
num  target     prot opt source               destination
1    DROP       all  --  0.0.0.0/0            0.0.0.0/0           ctstate INVALID
2    bad_tcp_packets  tcp  --  0.0.0.0/0            0.0.0.0/0
3    RETURN     all  --  0.0.0.0/0            0.0.0.0/0
```

- **Delete a Rule**
  - Allows you to select the Chain containing the rule you wish to delete, and the specific rule number you wish to delete

- **Add a Rule**
  - Allows for the addition of rules – see details below

Save Rules

- Save a copy of the current iptables ruleset in a directory of your choice. The file will be saved with a name of “iptables_yyyymmdd_hhmmss.txt

- **Restore Rules from Saved File**
    - Reload rules from a file saved previously. This will overwrite and replace any existing rules.
- **Custom File to Load on Boot**
    - If this option is chosen, the ruleset in the specified file will be applied when the system is booted. It does this by substituting the default ruleset generated by the puppy script /*etc*/init.d/rc.firewall with this specified ruleset.
    - A file named **zFWBuddy-loadrules.sh** is placed in your startup group which, and it will be executed ***after*** the default ruleset is initially loaded.
    - To ensure this script is executed ***after*** the default ruleset is loaded, the file name begins with the letter ‘z’ and you can also specify a period (in seconds) for the script to wait before executing. If necessary you can adjust this timing to ensure the script executes after everything else in the autostart folder.
- **Add to Menu**
    - If this option is chosen, a desktop file will be added to */usr/share/applications* and the puppy menus refreshed. FWBuddy will be listed in the Network category in the menu.

Adding a Rule

Format of a simple iptables command

iptables ***<type> <chain name> <rule sequence>*** -p ***<protocol>*** -s ***<source IP address /range>*** -d ***<destination IP address/range>*** -j ***<action to take>***

A firewall **rule** specifies criteria for a *packet* and a *target*.

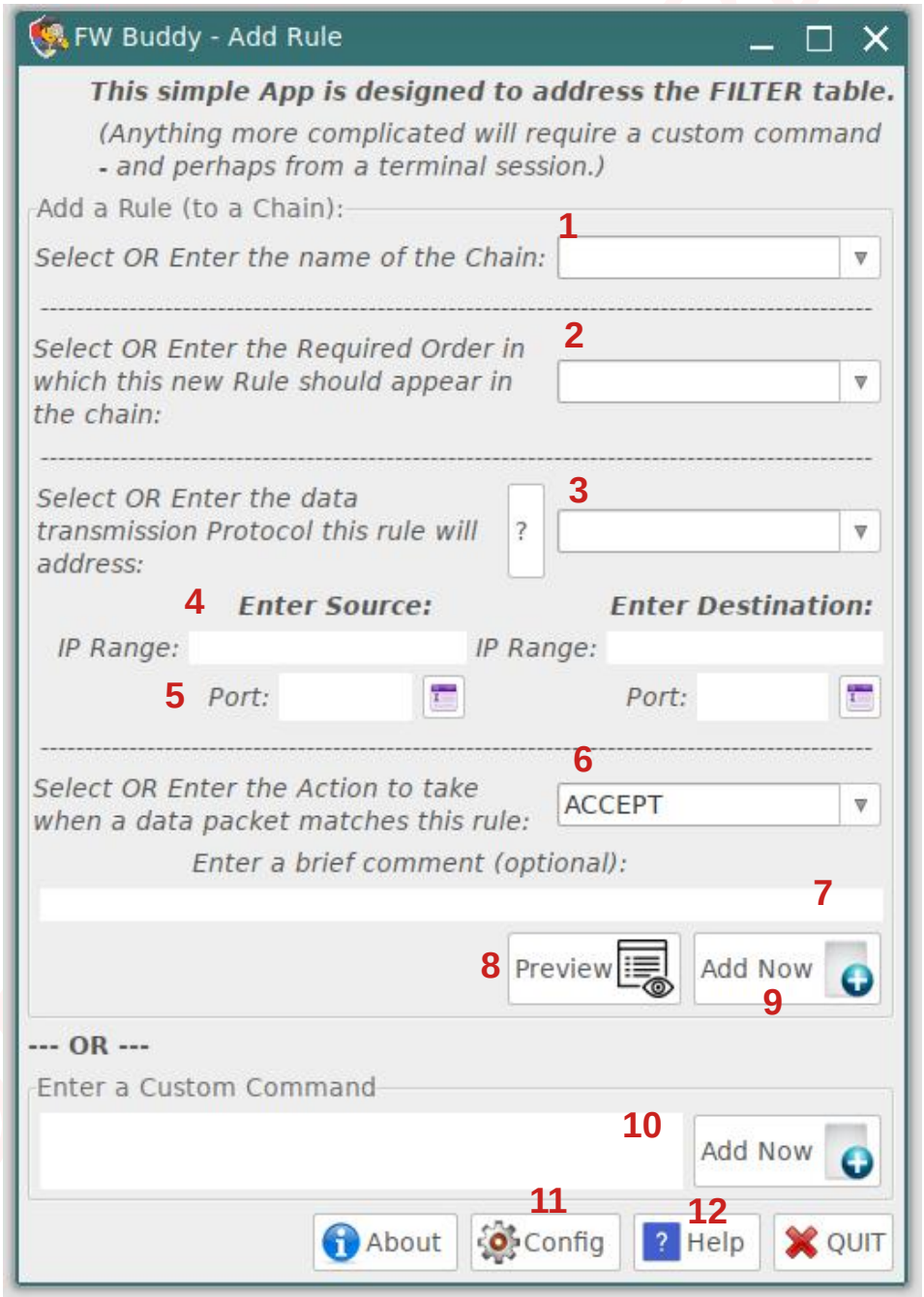
- If the packet does not match, the next rule in the chain is examined
- If it does match, then the packet is dealt with according to the 'target' (aka action) specified, which can be the name of a user-defined chain or one of the special values ACCEPT, DROP[,REJECT],QUEUE or RETURN.
  - ***ACCEPT*** *means to let the packet through.*
  - ***DROP*** *means to drop the packet on the floor, i.e. to discard it and not send any response*
  - ***REJECT*** *is used to send back an error packet in response to the matched packet: otherwise it is equivalent to DROP so it is a terminating TARGET, ending rule traversal.]*
  - ***QUEUE*** *means to pass the packet to user space.(ie the decision on what to do with the packet is delegated to a user space program. The user space software must use libnetfilter_queue to connect to queue 0 (the default one) and get the messages from kernel. It then must issue a verdict on the packet.)*
  - ***RETURN*** *means stop traversing this chain and resume at the next rule in the previous (calling) chain. If the end of a built-in chain is reached or a rule in a built-in chain with target RETURN is matched, the target specified by the chain policy determines the fate of the packet.*

Parameter	Explanation
<type>	Either: -I : to insert a rule between two already existing rules in a chain -A : to append a rule as the last entry in a chain
<chain name>	The name of the Chain to which the rule is to be added
<rule sequence>	The order of the rule in the Chain. **Order is very important**. If a data packet does not match a rule, then the next rule in the Chain is examined.
-p	Constant – signifies that the next parameter specifies a particular network protocol
<protocol>	The name of the data protocol. The most common will be: tcp udp icmp
-s	Constant – signifies that the next parameter specifies a 'source' network IP address or address range
<source IP address range>	The 'originator' IP address or address range of the data packet
-d	Constant – signifies that the next parameter specifies a 'destination' network IP address or address range
<destination IP	The 'target' IP address or address range of the data packet

address range>	
-j	Constant – signifies that the next parameter specifies the 'action' to take if the rule is matched
<action to take>	The action to take if the rule is matched. See above

Each of the above parameters can be (optionally) input to the "Add Rule" screen and the App will check the parameters for errors, and then translate and format them into a valid update command to iptables.

If the supplied parameters are not sufficient to create the rule required, a custom command can be entered in the space provided. This command will not be validated, but simply input to the iptables program exactly as described. ***Note: In this App a custom command cannot include comments.***



	Explanation
1	Select OR Enter the name of the chain. When you start the app it will interrogate your iptables and list each of the Chain currently defined. You can either select the required chain name from the drop down list, or enter it directly if you find this more efficient.
2	The order for the rule to be applied in the Chain. You may select from the drop down or enter directly.
3	The network protocol the rule should be applied to. You may select from the drop down list OR enter directly. Click on the ? Button to some of the most common protocols and ports.
4	Enter the source IP address AND the destination IP address. You may enter a range of addresses eg 192.168.1.0/24 If you want the rule to apply to ALL network addresses enter 0/0
5	Enter the source AND the destination Port. See (3) above to view some of the most common protocols and ports. If you want the rule to apply to ALL ports leave blank
6	The action to take when the rule is matched. You may select from the drop down list OR enter directly.
7	Enter a comment if you wish it to be attached to the rule
8	Click this button to 'translate' the parameters entered and check that the rule is what you want before applying it.
9	Click to apply the rule to iptables
10	If the supplied parameters are not sufficient to create the rule required, a custom command can be entered here. This command will not be validated, but simply input to the iptables program exactly as described. ***Note: In this App a custom command cannot include comments.***
11	Opens the 'config' file in your default text editor. You should not change any of these values unless you understand how the values are used.
12	Opens this User Manual in your default PDF viewer.