

Firewall Builder



The Problem

- In a heterogeneous environment, the administrator needs to be proficient with many different tools and CLI
- Administrator should understand how various firewalls differ in their capabilities and features
- Transition from one platform to another requires complete reconfiguration
- Good Open Source firewall implementations do not have decent user interface (iptables, ipfilter, pf)

What is Firewall Builder

- Multi-platform firewall configuration and management tool
- Administrator works with an abstract firewall rather than specific firewall implementation
- Uses object oriented approach to the firewall policy design
- Supports iptables, ipfilter, pf, ipfw and Cisco PIX
- Designed to support complex firewall configurations, can manage multiple firewalls from a single management workstation

Model of the Firewall

- Synthetic firewall, combines useful features found in different implementations and adds new ones
- Uses emulation to implement features absent on the target firewall platform
- Some features can not be emulated; the GUI and policy compilers can check for these limitations

Firewall Created by Firewall Builder

- policy and Network Address Translation (NAT) are represented as a set of standardized rules
- the first rule that matches the packet makes a decision and stops processing
- "implied deny" - empty policy blocks everything
- NAT is done before applying policy rules
- firewall assumed to be stateful
- policy rules may be associated with interfaces, but it is not mandatory
- negation is supported in policy and NAT rules

Supported Types of NAT rules

- translating source address and optionally a port
- translating destination address and optionally a port
- translating both source and destination address and optionally a port
- translating only port numbers
- "no nat" rule

GUI

- Uses object-oriented approach
- Presents objects and rules visually
- uses drag and drop operations to edit the policy
- Supports quick object inspection in rules using standard tooltip GUI widgets
- Comes with a library of standard objects
- Integrates network discovery wizards for quick object creation

GUI Screenshot

The screenshot shows the Firewall Builder application window. The title bar reads "Firewall Builder: /Users/vadim/Documents/DFN-CERT/objects.xml". The menu bar includes "File", "Edit", "View", "Insert", "Rules", "Tools", and "Help".

The left sidebar contains a tree view with the following items:

- User | Standard
- Name | Properties
- Objects
- Services
- Firewalls
 - firewall-pix
 - firewall
 - eth0 | 192.0.2.22 Ext
 - eth1 | 10.3.14.203 Mgr
 - eth2 | 10.2.2.1
 - Policy**
 - NAT
 - firewall
 - firewall2
- Time

The main area displays a table of firewall rules:

Num	Source	Destination	Service	Action	Options	Comment
00	Any	mail relay	TCP smtp	Accept		
01	mail relay	internal-net firewall	TCP smtp UDP dns	Accept		
02	mail relay	mail server	TCP smtp	Accept		
03	dmz-net	Any	Any	Deny		
04	internal-net	Any	Any	Accept		
05	Any	Any	Any	Deny		

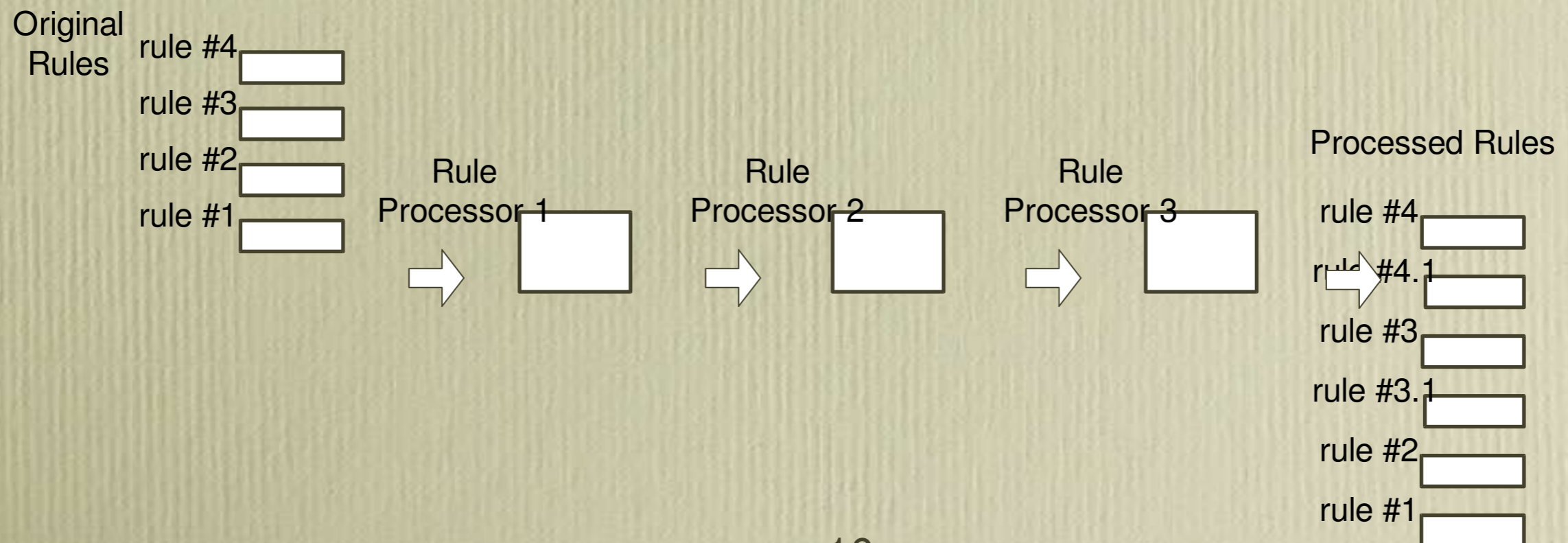
At the bottom right of the window, there are "Apply" and "Undo" buttons.

Standard Objects

- Objects and address ranges representing networks RFC 1918, “link-local”, “test net”, “this net”, multicast and broadcast addresses
- Over 130 objects representing often used services, such as http, smtp etc.
- Library is expanded with every release

Policy Compilers

- Translate rules defined in the GUI into the target firewall configuration language.
- Compiler consists of several elementary building blocks, or “Rule Processors”.
- Each rule processor performs elementary operation on a rule and passes it to the next.



Rule Processors

- Operations include rule verification, transformation and optimization.
- Rule processors may operate on a single rule or the whole rule set.
- Each rule processor is a C++ class
- Rule processors can be reused in different policy compilers







Detecting Errors

- It is important to catch as many errors in the policy as possible off-line
- Policy compilers recognize fatal and non-fatal errors:
 - misconfigurations of interfaces, addresses, netmasks
 - missing objects
 - Illegal and conflicting policy and NAT rules
 - "rule shadowing"

Examples of Rule Processors

1. Convert complex rule to a set of atomic rules
2. Translate rule with negation
3. Optimization

Example 1: A policy rule with many objects

Num	Source	Destination	Service	Action
00	 netA  netB	 hostC	 http  ftp	 Accept

If firewall does not support object grouping, this rule is expanded as follows:

Src	Dst	Srv	Action
netA	hostC	http	Accept
netA	hostC	ftp	Accept
netB	hostC	http	Accept
netB	hostC	ftp	Accept

Example 2: Policy Rule with Negation

Num	Source	Destination	Service	Action
00	 netA  netB	 hostC	 http	 Accept

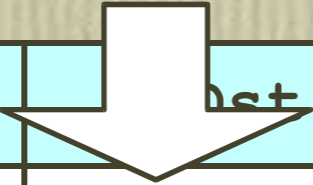
Many firewalls support negation in one of the rule elements, but the following simple translation is incorrect:

Src	Dst	Srv	Action
! netA	hostC	http	Accept
! netB	hostC	http	Accept

Example 2: Processed rule

Rule processor converts the rule:

Src	Dst	Srv	Action
! {netA, netB}	hostC	http	Accept



Src	Dst	Srv	Action
netA, netB	Any	Any	Continue
Any	hostC	http	Accept

Example 2: Generated Code

For iptables:






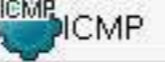

```
$IPTABLES -N TMPCHAIN  
$IPTABLES -A FORWARD -p tcp -d hostC --dport 80 -j TMPCHAIN  
$IPTABLES -A TMPCHAIN -s netA -j RETURN  
$IPTABLES -A TMPCHAIN -s netB -j RETURN  
$IPTABLES -A TMPCHAIN -m state --state NEW -j ACCEPT
```

For ipfilter:

```
skip 2 in proto tcp from netA to any  
skip 1 in proto tcp from netB to any  
pass in quick proto tcp from any to hostC port = 80
```

```
skip 2 out proto tcp from netA to any  
skip 1 out proto tcp from netB to any  
pass out quick proto tcp from any to hostC port = 80
```

Example 3: Optimization

Num	Source	Destination	Service	Action
00	 hostA  hostB	 net-1  net-2	 http  ICMP	 Accept

Trivial translation leads to $O(N^3)$ complexity:

Src	Dst	Srv	Action
hostA	net-1	http	Accept
hostA	net-1	icmp	Accept
hostA	net-2	http	Accept
hostA	net-2	icmp	Accept
hostB	net-1	http	Accept
hostB	net-1	icmp	Accept
hostB	net-2	http	Accept
hostB	net-2	icmp	Accept

Example 3: Optimization

Better translation of the same rule:

Chain	Src	Dst	Srv	Action
	hostA	Any	Any	C1
	hostB	Any	Any	C1
C1	Any	net-1	Any	C2
C1	Any	net-2	Any	C2
C2	Any	Any	http	Accept
C2	Any	Any	icmp	Accept

This has only $O(N)$ complexity

Conclusion

- Combines automation with flexibility, policy designer maintains full control
- Simplifies management of multiple firewalls in heterogeneous environments
- Provides easy migration path for different firewall platforms

The Project

- Started in 2000
- Hosted on SourceForge
- Home page: <http://www.fwbuilder.org/>
- Binary packages are built for
 - RedHat 7.3, 8.0, 9.0
 - Fedora Core 1
 - SuSE 8.2, 9.0
 - Mandrake 9.1
 - FreeBSD 4.9, 5.1
 - OpenBSD 3.4

Future Development

- Import objects and policy from existing firewall configuration
- Log analyzer
- Loadable policy templates
- Loadable object libraries
- Support for QoS and VPN
- Support for IPv6