Firewall Builder

Vadim Kurland
vadim@fwbuilder.org
http://www.fwbuilder.org

Challenges of firewall configuration

- complexity leads to errors
- coordinated changes on many devices are hard
- multi-vendor environments make the job even harder

- Many different kinds of devices - many different configurations
- Transition from one platform to another requires complete reconfiguration
-

What is Firewall Builder

- Creates configuration (iptables script, pf or pix configs)
- Currently supports iptables, ipfilter, pf, ipfw and Cisco IOS ACL and Cisco ASA (PIX)
- Administrator works with an abstract firewall rather than specific firewall implementation

There are many enterprise tools that manage configs, do audits, pushes etc. None of them help write configs in the first place.

What is Firewall Builder

- Uses object oriented approach to the firewall policy design
- Designed to support complex firewall configurations
- Can control multiple firewalls from a single management workstation
- Has built-in policy installer
- Has built-in revision control

Model of the Firewall

- Identify common principles of configuration languages and generalize
- If target device does not support a feature, emulate

Firewall Created by Firewall Builder

- A blend of features of iptables, PF and other
- always “first match”
- NAT before policy rules

policy and NAT are sets of standardized rules

“implied deny” - empty policy blocks everything

“first match” rule set

Use stateful rules if supported by target fw

object groups can be used in all rule elements

association of policy rules with interfaces is optional

NAT is done before policy rules

can use interfaces with dynamic address in rules via emulation if not supported natively (iptables)

Detecting Errors

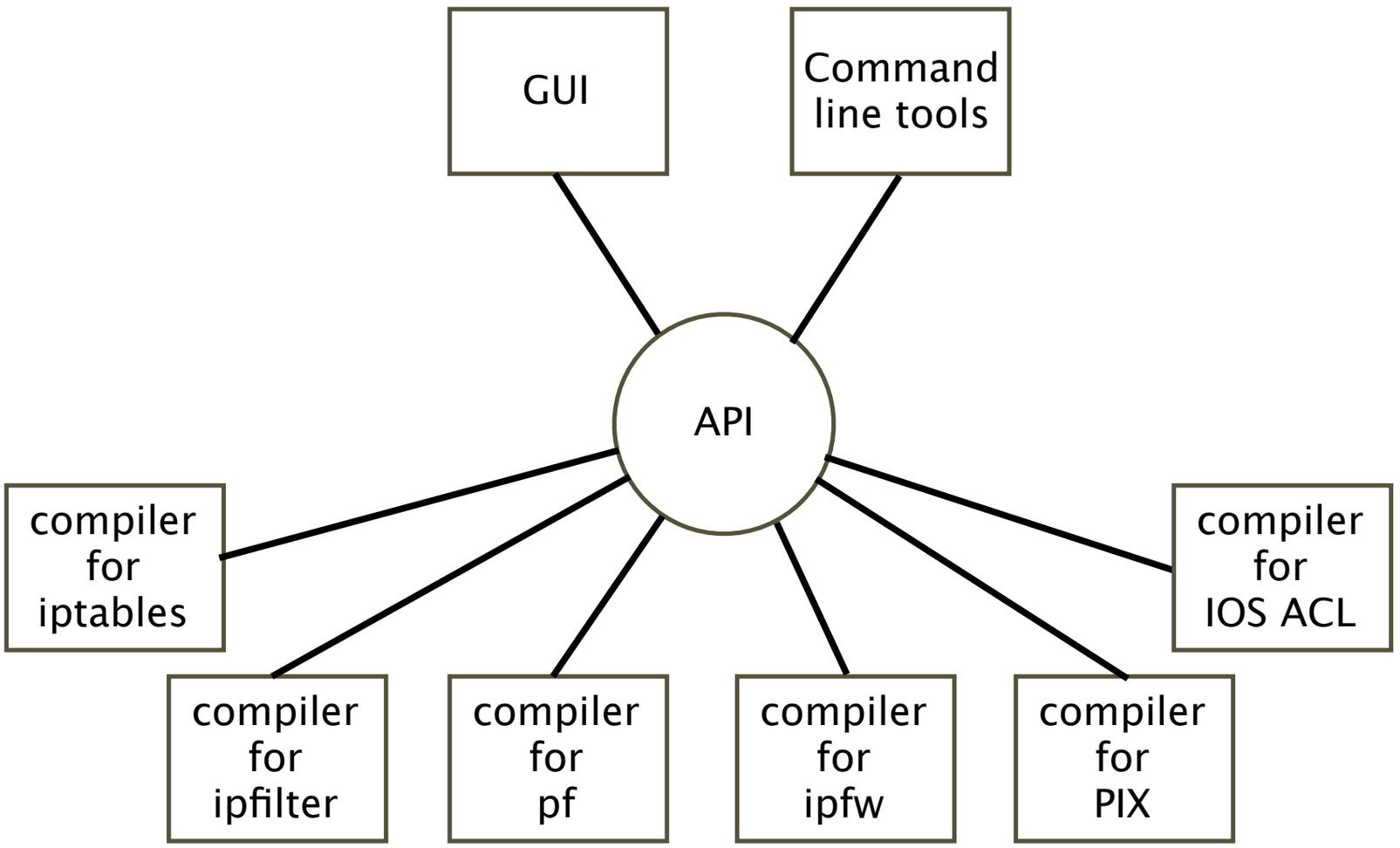
- Built-in “sanity checks” are aware of target platform limitations
- misconfigurations of interfaces, addresses, netmasks
- Illegal and conflicting policy and NAT rules
- "rule shadowing"

Invalid addresses and netmasks

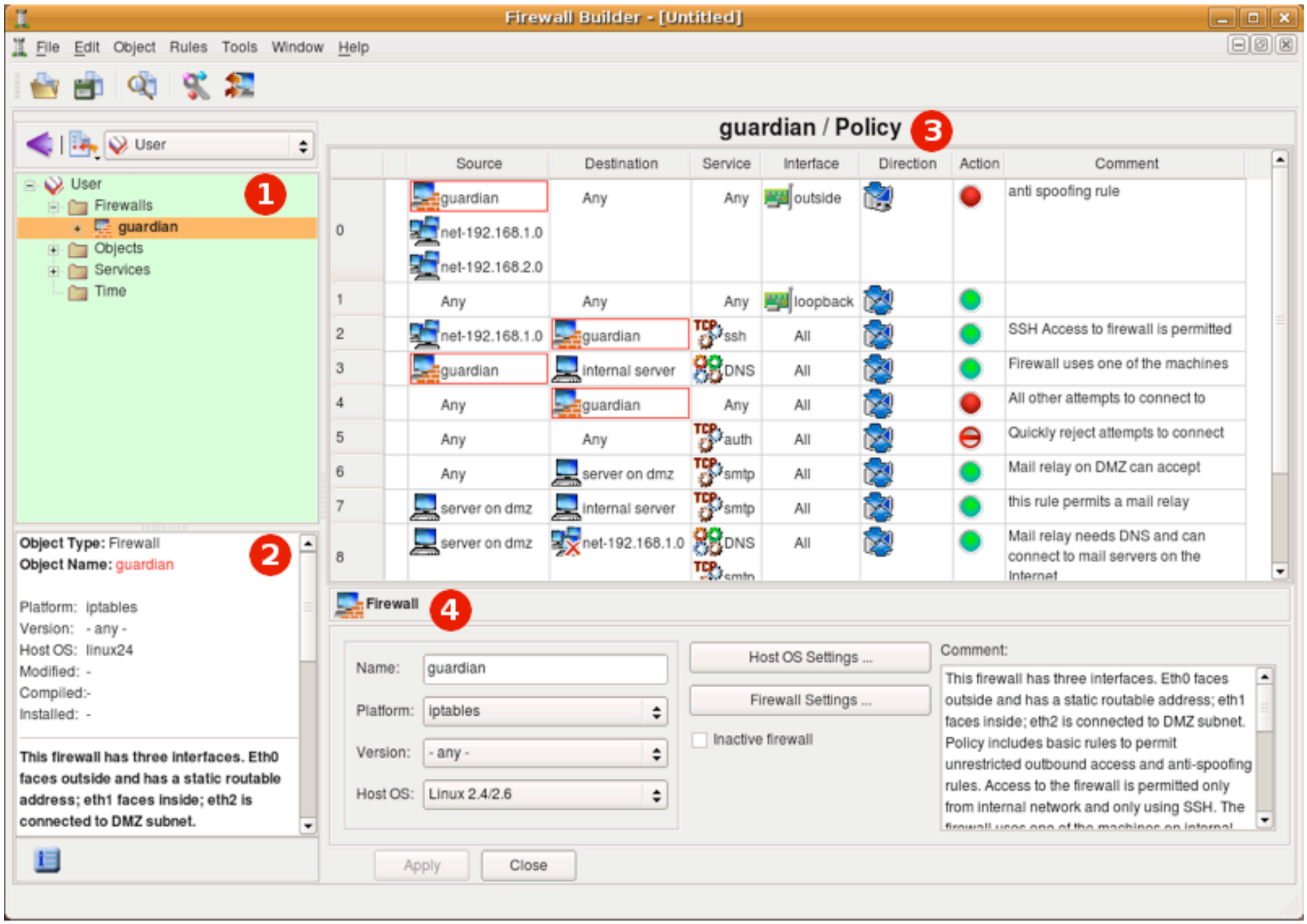
Invalid rules:
- MAC matching
- NAT: can’t translate tcp service into udp

Interfaces in the GUI don’t match real fw machine

Design and Implementation



GUI


Firewall Builder - [Untitled]
File Edit Object Rules Tools Window Help
User
User
Firewalls
guardian
Objects
Services
Time
1
Object Type: Firewall
Object Name: guardian
Platform: iptables
Version: - any -
Host OS: linux24
Modified: -
Compiled:-
Installed: -
This firewall has three interfaces. Eth0 faces outside and has a static routable address; eth1 faces inside; eth2 is connected to DMZ subnet.
2
guardian / Policy
3
Source
Destination
Service
Interface
Direction
Action
Comment
0 guardian net-192.168.1.0 net-192.168.2.0 Any Any outside anti spoofing rule
1 Any Any Any loopback
2 net-192.168.1.0 guardian ssh All SSH Access to firewall is permitted
3 guardian internal server DNS All Firewall uses one of the machines
4 Any guardian Any All All other attempts to connect to
5 Any Any auth All Quickly reject attempts to connect
6 Any server on dmz smtp All Mail relay on DMZ can accept
7 server on dmz internal server smtp All this rule permits a mail relay
8 server on dmz net-192.168.1.0 DNS smtp All Mail relay needs DNS and can connect to mail servers on the Internet
Firewall
4
Name: guardian
Platform: iptables
Version: - any -
Host OS: Linux 2.4/2.6
Host OS Settings ...
Firewall Settings ...
Inactive firewall
Comment:
This firewall has three interfaces. Eth0 faces outside and has a static routable address; eth1 faces inside; eth2 is connected to DMZ subnet. Policy includes basic rules to permit unrestricted outbound access and anti-spoofing rules. Access to the firewall is permitted only from internal network and only using SSH. The
Apply
Close

Example 1

NAT

Original Src	Original Dst	Original Srv	Translated Src	Translated Dst	Translated Srv
Any	outside	TCP ftp TCP smtp	Original	hostA	Original

Policy

Source	Destination	Service	Interface	Direction	Action
Any	hostA	TCP ftp TCP smtp	outside		

- Simple DNAT rule, access to the server “hostA” for two protocols: ftp and smtp
- Access from outside using address of interface “outside”
- Policy rule to permit ftp, smtp to the server

iptables

Original Src	Original Dst	Original Srv	Translated Src	Translated Dst	Translated Srv
Any	outside	TCP ftp TCP smtp	Original	hostA	Original

```
$IPTABLES -t nat -A PREROUTING  -p tcp -m tcp -m multiport \
   -d 192.0.2.1 --dports 21,25 \
   -j DNAT --to-destination 172.16.22.100
```

Source	Destination	Service	Interface	Direction	Action
Any	hostA	TCP ftp TCP smtp	outside		

```
$IPTABLES -A FORWARD  -i eth0 -p tcp -m tcp  -m multiport \
   -d 172.16.22.100  --dports 21,25  -m state --state NEW \
   -j ACCEPT
```

PF

Original Src	Original Dst	Original Srv	Translated Src	Translated Dst	Translated Srv
Any	outside	TCP ftp TCP smtp	Original	hostA	Original

```
rdr on en0 proto tcp from any to 192.0.2.1 port 21 -> \
    172.16.22.100 port 21
rdr on en0 proto tcp from any to 192.0.2.1 port 25 -> \
    172.16.22.100 port 25
```

Source	Destination	Service	Interface	Direction	Action
Any	hostA	TCP ftp TCP smtp	outside		

```
pass in   quick on en0 inet proto tcp \
    from any  to 172.16.22.100 port { 21, 25 }
```

PIX

```
class-map inspection_default
  match default-inspection-traffic

policy-map global_policy
  class inspection_default
    inspect ftp
    inspect esmtp

service-policy global_policy global

! Rule  0 (ethernet0)
!
access-list outside_acl_in  remark 0 (ethernet0)
access-list outside_acl_in permit tcp any host 192.0.2.1 eq 21
access-list outside_acl_in permit tcp any host 192.0.2.1 eq 25

access-group outside_acl_in in interface outside

! Rule  0 (NAT)
!
access-list id7036X25321.0 permit tcp host 172.16.22.100  eq 21 any
static (inside,outside) tcp interface 21  access-list id7036X25321.0 tcp 0 0
access-list id7036X25321.1 permit tcp host 172.16.22.100  eq 25 any
static (inside,outside) tcp interface 25  access-list id7036X25321.1 tcp 0 0
```

Example 2: A policy rule with many objects

Source	Destination	Service	Interface	Direction	Action
netA netB	hostC	ftp http	All	Both	Accept

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

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

Example 3: Policy Rule with Negation

Source	Destination	Service	Interface	Direction	Action
netA netB	hostC	http	All	Both	Accept

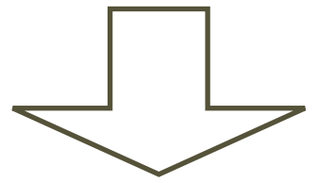
Many firewalls support negation in one of the rule elements, but the following simple translation is incorrect:

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

Example 3: Processed rule

The program converts the rule as follows:

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



Chain	Src	Dst	Srv	Action
FORWARD	Any	hostC	http	tmp_chain
tmp_chain	netA,netB	Any	Any	Return
tmp_chain	Any	Any	Any	Accept

Example 3: 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
```

Example 4: Optimization

Source	Destination	Service	Interface	Direction	Action
hostA hostB	net-1 net-2	http any ICMP	All	Both	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 4: 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

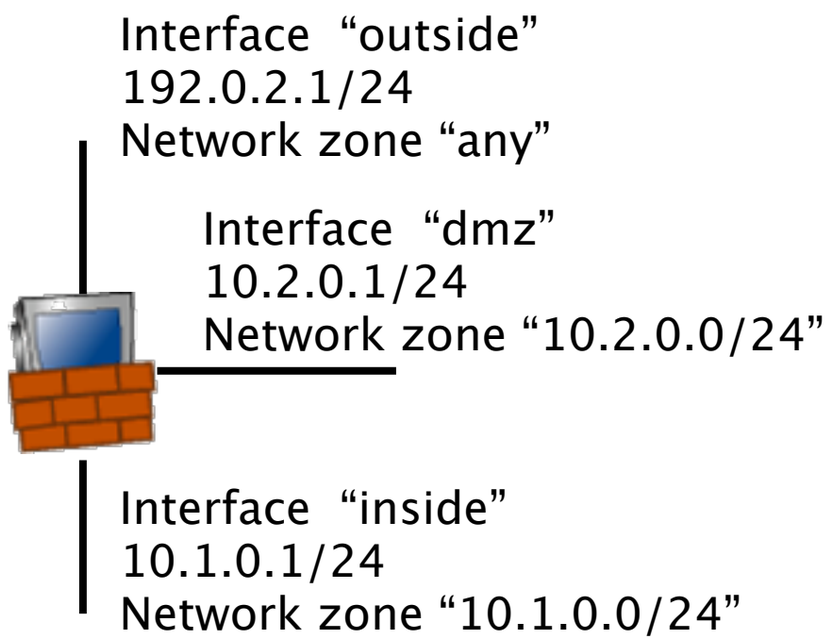
Example 5: Assigning Rules to Interfaces

- Some firewall can analyze packets regardless of ingress and egress interface, some can't
- In complex network configurations manual assigning rules to interfaces may be error-prone

Example 5: Assigning Rules to Interfaces

The following rules need to be assigned to interfaces:

Rule #	Src	Dst	
1	Any	10.2.0.10	..
2	10.2.0.1	10.1.0.10	..
3	10.1.0.1	10.2.0.10	..



Rule #1 is assigned to all interfaces
Rule #2 is assigned to interface “dmz”
Rule #3 is assigned to interface “inside”

When new network is added behind some interface, all you need to do is add it to the network zone of this interface and recompile. If there are rules that should be added to this interface because of the new network, the program will add them automatically.

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

Extras

Future Development

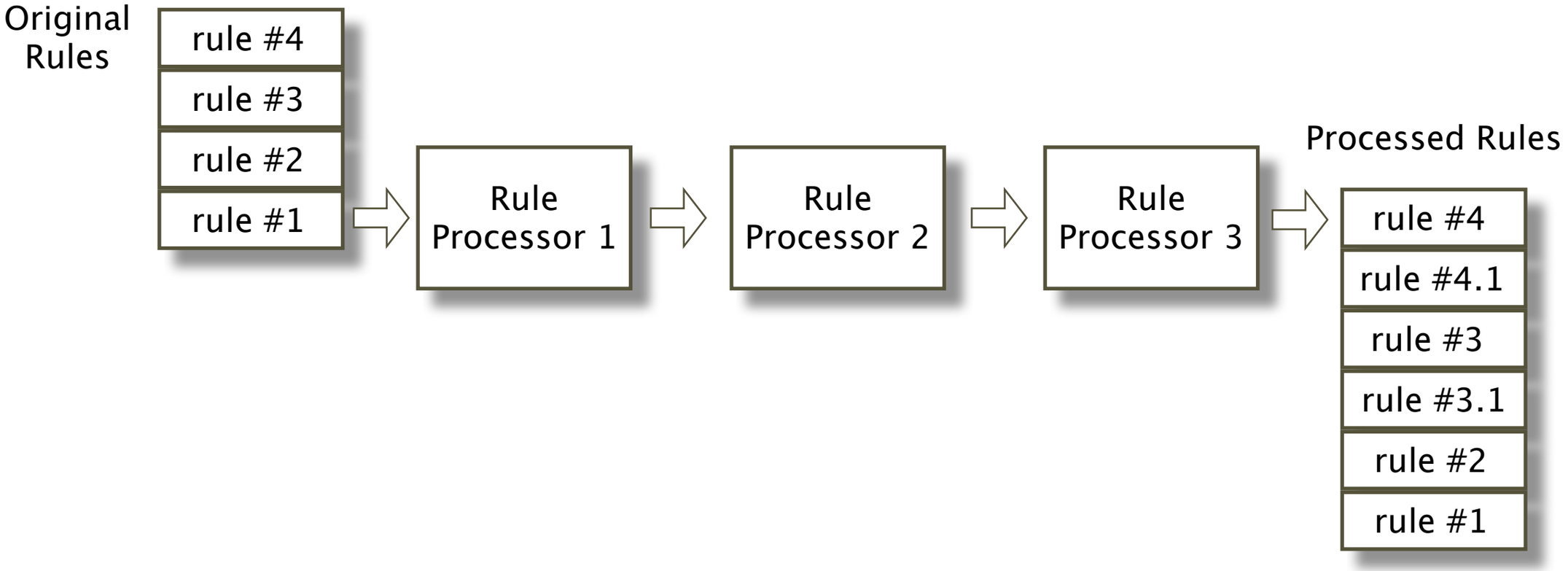
- High Availability configurations
- Support for QoS
- Templates with parameters
- Log analyser

The Project

- Started in 2000
- Hosted on SourceForge
- Home page: http://www.fwbuilder.org/
- Binary packages are built for
  - Fedora Core
  - Ubuntu
  - FreeBSD

Policy Compilers

- Translate rules defined in the GUI to 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 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

Examples of Rule Processors

- Convert complex rule to a set of atomic rules
- Translate rule with negation
- Optimization