How to manage iptables with puppet

Marc Fournier

18. february 2010

Hmmm...

Or maybe not ?

common approaches

- erb templates
- file fragments
- augeas
- `iptables-restore`

$\Rightarrow$ manage iptable backup file, then reload from backup

$\Rightarrow$ same needs, many different home-made solutions.

native iptables type

⇒ Dmitri Priimak, october 2007

```
iptables { "80":
    destination => "10.0.0.1",
    ensure      => "open",
}
```

- more flexibility
- rules added one by one, then old rules removed
- resources collected, then rules applied in one run
- suitable for simple firewalls (else use shorewall)

overview of hosts/network iptable rules

```
SELECT
    p.name, v.value, r.title, h.name
FROM
    param_names AS p,
    param_values AS v,
    resources AS r,
    hosts AS h
WHERE
    r.id=v.resource_id AND
    p.id=v.param_name_id AND
    r.host_id=h.id AND
    r.restype="Iptables" AND
    h.name="foobar.example.com";
```

$\Rightarrow$ better: use ActiveRecords

issues

- unusual terminology
- bug: firewall reloaded at every puppet run
- very few options

current version attempts to solve these issues.

⇒ `http://github.com/camptocamp/puppet-iptables`

examples

```
iptables { "allow icmp":
    proto => "icmp",
    icmp  => "any",
    jump  => "ACCEPT",
}
```

examples

```
iptables { "reject ssh from hostile subnet":
    source => "10.1.0.0/16",
    proto  => "tcp",
    dport  => "22",
    jump   => "REJECT",
    reject => "icmp-port-unreachable",
}
```

examples

```
iptables { "redirect port 80 to port 8080":
    chain    => "PREROUTING",
    table    => "nat",
    proto    => "tcp",
    dport    => "80",
    redirect => "8080",
    jump     => "REDIRECT",
}
```

the problem with iptables & puppet

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT

                          !=

iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

- puppet resource order is semi-random
- iptables resources order does really matter

the only reasonable solution

```
iptables { "a": }
iptables { "b": require => Iptables["a"] }
iptables { "c": require => Iptables["b"] }
iptables { "d": require => Iptables["c"] }
```

- inconvenient with more than a few rules
- a problem when rules are defined in modules

Help !

I need your opinion...