Securing and Extending Puppet for World Domination

Hi, I’m Richard Crowley

Equal opportunity technology hater

DevStructure’s operator and UNIX hacker

Configuration management Cliff's Notes

Infrastructure as code

You can reason about code in ways you can’t about a tarball or AMI.

Declare state, not process

More precise. Less verbose.

The state of a server is easier to unambiguously describe.

Why manage configuration?

An interlude on package management

Configuration management is the centralized authority to a package manager’s local authority.

Puppet, Chef, and `~/bin/doit5`

Limits are good.

Puppet and Chef are idempotent by default.

Architecture

Master knows best.

Agents phone home.

Hostname matching.

Resources.

Agents make it so.

Resources

The smallest unit of configuration.

```
package { "foo": ensure => "0.0.0" }

file { "/etc/foo.conf":
    content => template("foo.conf.erb"),
    owner   => "foo",
    mode    => "600",
    ensure  => file,
}
```

Classes and definitions

Compose resources in interesting ways.

```
class bar {
    exec { "apt-get update": }
    package { "bar":
        require => Exec["apt-get update"],
        ensure  => latest,
    }
    file { "/etc/bar.conf":
        content => template("bar.conf.erb"),
        ensure  => file,
    }
}
```

A basic configuration for Puppet itself

/etc/puppet/puppet.conf

```
[main]
    logdir=/var/log/puppet
    rundir=/var/run/puppet
    ssldir=$vardir/ssl
    pluginsync=true
    server=puppetmaster.example.com
```

/etc/cron.d/puppet

```
PATH="/usr/sbin:/usr/bin:/sbin:/bin"

# Remove the line breaks.
*/30 * * * * root bash -c '
    sleep $(($RANDOM \% 1800));
    puppet agent --certname=$(cat
    /etc/puppet/certname)
    --no-daemonize --onetime'
```

(R)TFM

```
puppet --genconfig
```

http://bit.ly/puppet-config-ref

Hello, world!

Entire thing in `/etc/puppet`

Entrypoint is `manifests/site.pp`

Hello, `manifests/site.pp`!

```
import "nodes"

Exec {
    path => "/usr/sbin:/usr/bin:/sbin:/bin",
}
```

Hello, `manifests/nodes.pp`!

```
node default { include base }

node www inherits default { include www }

node 'staging.example.com' inherits www {}
node /\.www\.example\.com$/ inherits www {}
```

Hello, base!

modules/base/manifests/init.pp

```
class base {
    package {
        "dnsutils": ensure => latest;
        "psmisc": ensure => latest;
        "strace": ensure => latest;
        "sysstat": ensure => latest;
        "telnet": ensure => latest;
    }
}
```

Hello, `www`!

`modules/www/manifests/init.pp`

```
class www {
    package { "nginx":
        ensure => "0.7.65-1ubuntu2",
    }
}
```

Get yours

http://bit.ly/devstructure-puppet-agent

http://bit.ly/devstructure-puppet-master

Security

SSL Cliff’s Notes

SSL handshake

Server certificate authenticates server to client.

Clients may verify a server’s certificate against trusted certificate authority.

Client and server compute matching secret keys.

Client certificates

Client certificate authenticates client to server.

Not used by browsers.

Used by Puppet because you can’t lie here.

SSL in Puppet

`/var/lib/puppet/ssl` on agents.

`puppet cert` on master.

http://bit.ly/puppet-security-ref

A tour of secondary Puppet config files

autosign.conf

```
foo.example.com
*.www.example.com
*
```

A bad idea when there are untrusted clients.

auth.conf

```
path ~ ^/catalog/([^/]+)$
method find
allow $1
```

Safe by default. Easy to mistakenly open.

http://bit.ly/puppet-security-ref

http://bit.ly/puppet-rest-ref

fileserver.conf

```
[modulename]
    path /foo/bar/baz
    allow *
```

Usually certificates have to be signed.

http://bit.ly/puppet-fileserver-ref

Why to lie to Puppet

Gather compromising information about other servers

`iptables` rules.

Database passwords.

SSH private keys.

`/etc/passwd` entries.

AWS credentials.

How to lie to Puppet

How to lie to Puppet

Maybe they're `autosign`ing?

How to lie to Puppet

Sneak through regex `node` definitions.

```
[agent]
    certname=foo.www.example.com
```

How to lie to Puppet

Unmatched names fall back to `default`.

```
[agent]
    certname=adhadgsdhsfsdhxcb.example.com
```

How to lie to Puppet

`autosign` disabled? Try social engineering.

```
[agent]
    certname=admin0.example.com
    # certname=dev.example.com
    # certname=test.example.com
    # certname=corp.example.com
```

How to lie to Puppet

With a signed certificate, snoop for other catalogs.

```
export ssldir=/var/lib/puppet/ssl
export server=puppetmaster.example.com

curl --insecure \
    --cert $ssldir/certs/$CERTNAME.pem \
    --key $ssldir/private_keys/$CERTNAME.pem \
    --cacert $ssldir/ca/ca_crt.pem \
    https://$server:8140/$ENV/catalog/db1.example.com
```

How to lie to Puppet

Snoop around an open file server, no certificates needed.

```
export server=puppetmaster.example.com

curl --insecure \
    https://$server:8140/$ENV/file_content/$MODULE/$FILE
```

How to safely not care

Don’t autosign anything, ever.

How to safely not care

~~Don’t autosign anything, ever.~~

Understand and protect the network.

Remove special cases.

The network

Understand who can contact your Puppet master.

Private networks may be shared.

iptables

```
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

iptables -F

iptables -A INPUT -m conntrack \
    --ctstate RELATED,ESTABLISHED -j ACCEPT

iptables -A INPUT -i eth1 -p tcp \
    -s 10.47.0.0/16 --dport 8140 -j ACCEPT
iptables -A INPUT -i eth1 -p udp \
    -s 10.47.0.0/16 --dport 8140 -j ACCEPT

iptables -A INPUT -i lo -j ACCEPT

iptables -A INPUT -j DROP
```

`stunnel`

(Not specific to Puppet.)

If it's on the Internet,
it's either public or encrypted.

Example: use `stunnel`(8)
to make `$NoSQL` SSL-aware.

`stunnel` Upstart config

```
description "stunnel-redis-client"
start on runlevel [2345]
stop on runlevel [!2345]
respawn
exec /usr/bin/stunnel -f -c -d localhost:6379 \
    -r redis.example.com:6381

description "stunnel-redis-server"
start on runlevel [2345]
stop on runlevel [!2345]
respawn
exec /usr/bin/stunnel -f -d 6381 -r localhost:6382
```

Special cases aren’t that special

Use templates

```
file { "/foo/bar/baz":
    ~~source  => "puppet://foo/bar/baz",~~
    content => template("foo/bar/baz"),
    ensure  => file,
}
```

File and catalog serving use different ACLs.

Template content is included in the catalog so they inherit the catalog's ACL.

Use only the `default` node

Mitigate the consequences of successful `certname` speculation.

Use `--config`, `--manifestdir`, or `--manifest` to run different masters listening on different interfaces/ports.

Extending Puppet

Maybe don't

```
$extlookup_datadir = "/etc/puppet/extdata"
$extlookup_precedence = [
    "%{fqdn}", "%{domain}", "base"]

file { "/foo/bar/baz":
    content => extlookup("foobarbaz"),
    ensure  => file,
}
```

`extlookup` function can retrieve data from external CSV files.

Then again, maybe extend Puppet

Where does your code run?

External node classifier runs on master.

Plugins run on agents.

Example task

Authorize an SSH key unique to each host.

(This is by no means impossible in the Puppet language.)

External node classifier

Run your own code on the master.

Make decisions based on
more than just the hostname.

Configuring an external node classifier

```
[master]
    external_nodes=/usr/local/bin/classifier
    node_terminus=exec
```

Input

Hostname...

...which maps to facts.

```
/usr/local/bin/classifier foo.example.com

# Facts are available as YAML in
# $vardir/yaml/facts/foo.example.com.yaml
```

Facts?

Key value pairs describing the server in question.

```
--- !ruby/object:Puppet::Node::Facts
  expiration: 2010-09-20 20:27:14.445807
  name: &id003 foo.example.com
  values:
    hardwaremodel: &id002 x86_64
    kernelrelease: 2.6.35.1-rscloud
    selinux: "false"
    sshrsakey: OH HAI
```

Lots more: `facter | less`

Output

```
---
classes:
  - base
  - www
environment: production
parameters:
  mail_server: mail.example.com
```

Classes, variables. No resources. YAML.

Example external node classifier

```
#!/bin/sh
set -e
TMP=$(mktemp -d "$1.XXXXXXXXXX")
ssh-keygen -q -f "$TMP/id_rsa" -b 2048 -N ""
zomg_post_to_the_api "$1" "$(cat "$TMP/id_rsa")"
cat <<EOF
---
classes:
  - ssh
parameters:
  public_key: $(cat "$TMP/id_rsa.pub")
EOF
```

Example Puppet class

```
class ssh {
    file {
        "/root/.ssh":
            mode   => "700",
            ensure => directory;
        "/root/.ssh/authorized_keys":
            content => "$public_key\n",
            ensure  => file;
    }
}
```

Puppet plugins

Because sometimes Puppet won’t let you.

The easy way

"It's just Ruby."

The orderly way

Gather the necessary data.

Build a Puppet catalog complete with dependency declarations.

Apply the catalog.

Plugin file structure

```
modules/ssh/
    manifests/
        init.pp
    lib/puppet/
        type/
            keygen.rb
        provider/keygen/
            posix.rb
```

Types versus providers

A `package` is a type.
`apt` is a provider of packages.

Types parse and normalize arguments.
Providers make it so.

Manifest

```
class ssh {
    keygen { "name-it-whatever": }
}
```

Type

```ruby
require 'puppet/type'
Puppet::Type.newtype(:keygen) do
  @doc = "ssh-keygen example"
  newparam(:whatever, :namevar => true) do
    desc "Name it whatever."
  end
  ensurable do
    self.defaultvalues
    defaultto :present
  end
end
```

Provider

```
require 'openssl'
require 'puppet/resource'
require 'puppet/resource/catalog'

Puppet::Type.type(:keygen).
  provide(:posix) do

  desc "ssh-keygen example for POSIX"
  defaultfor :operatingsystem => :debian

  # Define exists?, create, and destroy.

end
```

```ruby
def exists?
  File.exists?(
    "/root/.ssh/authorized_keys")
end

def destroy
  Puppet.warning "No turning back."
  raise NotImplementedError
end
```

```ruby
def create
  key = OpenSSL::PKey::RSA.generate(2048)
  zomg_post_to_the_api \
    Facter.value(:certname), key
  catalog = Puppet::Resource::Catalog.new
  catalog.create_resource(:file,
    :path => "/root/.ssh",
    :mode => "700",
    :ensure => :directory
  )
  catalog.create_resource(:file,
    :path => "/root/.ssh/authorized_keys",
    :content => "#{key.public_key}\n",
    :ensure => :file
  )
  catalog.apply
end
```

Which is right? Easy or orderly?

That’s an exercise for the reader.

Extending Puppet

(One more thing.)

Rack middleware

Puppet master is Rack application.

Rack allows pre- and post- processing.

config.ru

```ruby
$0 = "master"
ARGV << "--rack"
ARGV << "--certname=#{
  File.read("/etc/puppet/certname").chomp}"

require 'puppet/application/master'

# TODO Middleware.

run Puppet::Application[:master].run
```

No-op middleware

```ruby
require 'base64'
require 'json'
require 'rack/utils'
require 'yaml'
require 'zlib'

class StuckInTheMiddleWithYou

  def initialize(app)
    @app = app
  end

  def call(env)
    # TODO Preprocessing.
    status, headers, body = @app.call(env)
    # TODO Postprocessing.
    [status, headers, body]
  end

end

use StuckInTheMiddleWithYou
```

Preprocessing

```
params = Rack::Utils.parse_query(env["QUERY_STRING"], "&")
facts = case params["facts_format"]
when "b64_zlib_yaml"
  YAML.load(Zlib::Inflate.inflate(Base64.decode64(
    Rack::Utils.unescape(params["facts"]))))
end

# Change facts.
if Puppet::Node::Facts === facts
  facts.values["foo"] = "bar"
end

params["facts"] = case params["facts_format"]
when "b64_zlib_yaml"
  Rack::Utils.escape(Base64.encode64(Zlib::Deflate.deflate(
    YAML.dump(facts), Zlib::BEST_COMPRESSION)))
end if facts
env["QUERY_STRING"] = Rack::Utils.build_query(params)
env["REQUEST_URI"] =
  "#{env["PATH_INFO"]}?#{env["QUERY_STRING"]}"
```

Postprocessing

```
object = case headers["Content-Type"]
when /[\/-]pson$/ then JSON.parse(body.body.join)
when /[\/-]yaml$/ then YAML.load(body.body.join)
when "text/marshal" then Marshal.load(body.body.join)
else body.body.join
end

# Change catalog.
if Hash === object && "Catalog" == object["document_type"]
  object["data"]["resources"].unshift({
    "exported" => false,
    "title" => "apt-get update",
    "parameters" => {"path"=>"/usr/sbin:/usr/bin:/sbin:/bin"}
    "type" => "Exec",
  })
end

body = case headers["Content-Type"]
when /[\/-]pson$/ then [JSON.generate(object)]
when /[\/-]yaml$/ then [YAML.dump(object)]
when "text/marshal" then [Marshal.dump(object)]
else [object]
end
headers["Content-Length"] = Rack::Utils.bytesize(body.first)
```

Rack middleware

Drink responsibly.

http://gist.github.com/602922

Thank you

richard@devstructure.com or @rcrowley

http://rcrowley.org/talks/puppet-camp-2010

P.S. use DevStructure.