去中心化puppet部署

huangmingyou@gmail.com

December 17, 2013

目录

前言

去中心化的puppet部署，舍弃puppetmaster，直接把puppet manifest 部署到puppet客户端进行执行。这样部署能带来两大好处，一是无穷的扩展性能，c/s方式部署的puppet,部署规模一大，puppetmaster性能会成为一个瓶颈。 二是增加安全性，c/s模式下的证书认证， 只能保证数据在传输过程中的安全性，但是一旦puppetmaster被黑，所有puppet agent也面临被黑的风险。

本手册的最新版本和tex源代码,以及手册里面列出的几个脚本可以从:http://github.com/huangmingyou/puppet/获得。

第1章　部署

《草》

作者：白居易

離離原上草，一歲一枯榮。

野火燒不盡，春風吹又生。

遠芳侵古道，晴翠接荒城。

又送王孫去，萋萋滿別情。

1.1　整个系统的结构

核心思路，把puppet manifest 打包，通过gnupg 签名以后，通过任何手段传输到 puppet agent上执行。 这样带来的意义何在？

首先，不用puppet master来解释和分发代码，没有了单点负载压力，因为你可以通过ftp,rsync,https,等各种传输手段来分发puppet 的manifest到puppet agent。你甚至可以考虑用cdn来分发。当然，这样传输密码等关键信息是不行的，你完全可以设计另一条安全的路径来传输机密信息,比如scp。以我生产环境来说，我是用rsync来传输的，因为我的代码里面没有机密信息。也不怕公开。

其次，利用gpg对代码签名来保证puppet agent执行的代码是经过确认的安全代码，puppet agent上的gpg 公钥可以在安装系统的时候初始化安装。这样的安全性高于主流的c/s puppet部署方式。 c/s 部署的证书作用有两个，一是防止假的puppet agent 来puppet master 骗取puppet 配置。 二是作为https传输的证书。仅此而已！ 如果你的企业有上千台的机器部署了 c/s 模式的puppet. 那么所有的安全都系在puppet master的安全上了。一旦puppet master沦陷，所有机器沦陷。这都是因为puppet缺少一个puppet manifest代码的审核机制。只要puppet代码没有语法错误，puppet master就会解析执行并传送给puppet agent执行。总的说来，puppet还是缺少授权和对代码的认证。

利用gpg签名puppet manifest代码，安全性能提高不少，因为，只要保护好gpg私钥和密码，就能保证puppet agent执行的代码不是被篡改过的代码。 gpg

私钥可以通过保存到网络隔离的机器上来保证安全。并且做磁盘加密。能做到不错的安全程度,加密puppet manifest 代码的时候，利用u盘来拷贝。而puppet master很难做到网络隔离,网络都断了，还怎么和puppet agent通讯。

但是，没有绝对的安全！毕竟，太阳也有毁灭的一天。

1.2 部署实例

以我当前的生产环境的部署来作为例子，我利用一台最垃圾的pc作为私钥保存和签名的机器，并且网络隔离，做磁盘加密。利用rsync来分发代码。我们来看看这套系统怎样自己提着自己的鞋带把自己拉起来。

当运维上线一台新机器的时候，加入我们自己的私有软件仓库，然后用 apt-get 安装 xy-puppet-init包。这个包依赖puppet,会自动把puppet安装好。同时这个包会在/etc/cron.d/里面安装一个定时任务。这个定时任务会每隔一小时(具体是一小时中那一分钟，是安装包的时候随机设置的,防止所有客户端同时下载。)从rsync server去下载puppet manifest代码来执行。下面就是这个脚本的内容。

```
#!/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
[ -f /nopuppet ]&&exit 0
rsync -avz --delete --exclude='.svn' puppet@rsyncserver::puppet/ /opt/puppet/
[ -d /tmp/xy-puppet ]||mkdir /tmp/xy-puppet
rm /tmp/xy-puppet/* -rf
tar zxf /opt/puppet/puppet.tgz -C /tmp/xy-puppet
gpg --verify puppet.tgz.asc
[ $? -ne 0 ]&&exit 0
rsync -avz --delete /tmp/xy-puppet/puppet/ /etc/puppet/
puppet /etc/puppet/manifests/site.pp
/bin/run-parts /etc/puppet/file/shell/autorun/
```

脚本每隔一小时从rsync服务器下载puppet.tgz 和签名文件。如果签名正确，就让puppet 执行/etc/puppet/manifest/site.pp文件。我的生产环境里面，是用eth0的ip地址作为hostname来区别不同的机器的。这样执行puppet代码，也可以利用template,变量等常用的puppet特性。外部资源和虚拟资源不能用。

第2章　信息收集

題目:《柳》

作者:寇準（９６１—１０２３）

曉帶輕烟間杏花，晚凝深翠拂平沙。

長條别有風流處，密映錢塘蘇小家。

2.1　信息收集

如何收集puppet执行后的信息呢? 我这里是采用的foreman来收集puppet执行情况以及服务器上的facter配置信息。通常，在c/s方式部署puppet的时候，puppet agent执行的结果反馈给puppet master, 然后由puppet master统计提交给foreman或者puppet dashboard。在这里，因为没有puppet master,所以我们需要直接把puppet执行后情况发送给foreman。思路是在puppet agent的puppet.conf配置文件里面加上log report功能，然后自定义一个foreman的log report方法。

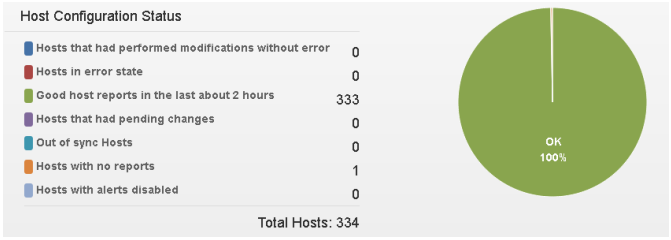
```
[main]
report = true
reports = foreman
```

然后再创建一个/var/lib/puppet/lib/puppet/reports/foreman.rb的文件，内容如下:

```
# <%= ERB.new(File.read(File.expand_path("_header.erb",File.dirname(file)))).result(binding) -%>
# copy this file to your report dir - e.g. /usr/lib/ruby/1.8/puppet/reports/
# add this report in your puppetmaster reports - e.g, in your puppet.conf add:
# reports=log, foreman # (or any other reports you want)

# URL of your Foreman installation
$foreman_url='http://foremanserver:8000'
# if CA is specified, remote Foreman host will be verified
#$foreman_ssl_ca = "<%= @ssl_ca -%>"
# ssl_cert and key are required if require_ssl_puppetmasters is enabled in Foreman
#$foreman_ssl_cert = "<%= @ssl_cert -%>"
#$foreman_ssl_key = "<%= @ssl_key -%>"

require 'puppet'
require 'net/http'
require 'net/https'
require 'uri'

Puppet::Reports.register_report(:foreman) do
    Puppet.settings.use(:reporting)
```

```
    desc "Sends␣reports␣directly␣to␣Foreman"

    def process
      begin
        uri = URI.parse($foreman_url)
        http = Net::HTTP.new(uri.host, uri.port)
        http.use_ssl     = uri.scheme == 'https'
        if http.use_ssl?
          if $foreman_ssl_ca && !$foreman_ssl_ca.empty?
            http.ca_file = $foreman_ssl_ca
            http.verify_mode = OpenSSL::SSL::VERIFY_PEER
          else
            http.verify_mode = OpenSSL::SSL::VERIFY_NONE
          end
          if $foreman_ssl_cert && !$foreman_ssl_cert.empty? && $foreman_ssl_key && !$foreman_ssl_key.empty?
            http.cert = OpenSSL::X509::Certificate.new(File.read($foreman_ssl_cert))
            http.key  = OpenSSL::PKey::RSA.new(File.read($foreman_ssl_key), nil)
          end
        end
        req = Net::HTTP::Post.new("#{uri.path}/reports/create?format=yml")
        req.set_form_data({'report' => to_yaml})
        response = http.request(req)
      rescue Exception => e
        raise Puppet::Error, "Could␣not␣send␣report␣to␣Foreman␣at␣#{$foreman_url}/reports/create?format=yml:␣#{e}"
      end
    end
end
```

这个脚本会把*puppet*执行日志发送给*foreman*。这样就可以知道*puppet*执行情况。



另外还有一个数据的收集，那就是*facter*收集的数据，便于做资产统计，比如统计服务器序列号，软件的版本号，网卡的速率等。下面这个脚本，会把*/var/lib/puppet/yaml/facts/hostname.yaml*文件内的内容，推送到*foreman*里面的*fact*值里面。

```
#! /usr/bin/env ruby
#
# This scripts runs on remote puppetmasters that you wish to import their nodes facts into Foreman
# it uploads all of the new facts its encounter based on a control file which is stored in /tmp directory.
# This script can run in cron, e.g. once every minute
# if you run it on many puppetmasters at the same time, you might consider adding something like:
# sleep rand(10) # that not all PM hammers the DB at once.
# ohadlevy@gmail.com

# puppet config dir
puppetdir="/var/lib/puppet"

# URL where Foreman lives
url="http://122.224.73.164:8000"

# Temp file keeping the last run time
stat_file = "/tmp/foreman_fact_importer"

require 'fileutils'
require 'net/http'
require 'net/https'
require 'uri'
```

```
last_run = File.exists?(stat_file) ? File.stat(stat_file).mtime.utc : Time.now – 365*60*60

Dir["#{puppetdir}/yaml/facts/*.yaml"].each do |filename|
  last_fact = File.stat(filename).mtime.utc
  if last_fact > last_run
    fact = File.read(filename)
    puts "Importing␣#{filename}"
    begin
      uri = URI.parse(url)
      http = Net::HTTP.new(uri.host, uri.port)
      if uri.scheme == 'https' then
        http.use_ssl = true
        http.verify_mode = OpenSSL::SSL::VERIFY_NONE
      end
      req = Net::HTTP::Post.new("/fact_values/create?format=yml")
      req.set_form_data({'facts' => fact})
      response = http.request(req)
    rescue Exception => e
      raise "Could␣not␣send␣facts␣to␣Foreman:␣#{e}"
    end
  end
end
FileUtils.touch stat_file
```

现在，有趣的的问题来了，*/var/lib/puppet/yaml/facts/hostname.yaml* 文件怎么创建呢，*c/s*模式里面，是*puppetmaster*负责的。这里，我们直接用*facter –y* 命令来产生这个文件。这还产生了一个另外好处，不用自定义*facter*的脚本，就能收集任何*fact*值，比如我要收集一个软件版本的信息，我直接用*shell*脚本取软件版本号，然后用*echo*把信息追加到*yaml*文件里面。下面这个脚本我用来生成*yaml*文件。并且收集了几个软件的版本和网卡的速率。

```
#!/bin/bash
cf=/var/lib/puppet/yaml/facts/`hostname`.yaml
mkdir –p /var/lib/puppet/yaml/facts
facter –y  > $cf
sed –i '/ppp/d' $cf

m2v=$(dpkg –l|grep mycomp–l2tpns|awk '{print␣$3}')
m3v=$(dpkg –l|grep mycomp–openvpn|awk '{print␣$3}')
m2id=$(grep "id=" /opt/mycomp/l2tpns/etc/vppp.conf |sed 's/id=//')
echo "␣␣m2ver:␣${m2v:–NULL}" >>$cf
echo "␣␣m3ver:␣${m3v:–NULL}" >>$cf
echo "␣␣m2id:␣${m2id:–NULL}" >>$cf


mii–tool 2>/dev/null |sed 's/^/␣␣mii/' >>$cf
/usr/local/bin/push–fact.ruby
```

Name	Value	Reported at
miieth2	negotiated 100baseTx-FD, link ok	N/A

这个脚本会在每次执行完*puppet*以后调用。把需要收集的*fact*值发到*foreman*。

如果你思路还清晰的话，应该记的，每次都是从*crontab*执行脚本来下载*puppet*代码执行，这个脚本最后一行会执行*/bin/run–parts /etc/puppet/file/shell/autorun/*；会把 */etc/puppet/file/shell/autorun/* 目录下的所有脚本执行一次。有个好处，避免每次用*exec*资源来执行脚本。同时我还可以加一个*/etc/puppet/file/shell/perhost/* 目录，目录下面按主机名建目录，里面放上特定的

机器需要执行的特定代码。只需要再加一句行/bin/run-parts /etc/puppet/file/shell/perhost/`hostname`/

利用autorun和perhost目录，可以灵活的做一些批量任务，而不用去修改puppet代码，写exec资源等。