windows web server management with asp.net

#ChefConf 2014

Paul Oremland
http://paul.oremland.net
http://tech.infospace.com
https://github.com/poremland

the differences
chef on windows vs. linux

- cookbooks
  - still playing catchup
  - most things will require the windows cookbook
    - chef 11 added support for registry keys, powershell, batch scripts, and installing chef client as service
- idempotency
  - powershell idempotency can be built into the script or, whereever possible, can be done with the not_if/only_if guards
  - windows_batch, windows_zipfile, execute, and other dynamic behavior require the not_if/only_if guards for idempotency
- reboots
  - enabling/disabling certain windows features during a chef run may require a mid-run reboot; ex: UAC

the differences
chef on windows vs. linux

- cookbooks
  - still playing catchup
  - most things will require the windows cookbook
    - chef 11 added support for registry keys, powershell, batch scripts, and installing chef client as service
- idempotency
  - powershell idempotency can be built into the script or, whereever possible, can be done with the not_if/only_if guards
  - windows_batch, windows_zipfile, execute, and other dynamic behavior require the not_if/only_if guards for idempotency
- reboots
  - enabling/disabling certain windows features during a chef run may require a mid-run reboot; ex: UAC

the differences
chef on windows vs. linux

- cookbooks
  - still playing catchup
  - most things will require the windows cookbook
    - chef 11 added support for registry keys, powershell, batch scripts, and installing chef client as service
- idempotency
  - powershell idempotency can be built into the script or, whereeever possible, can be done with the not_if/only_if guards
  - windows_batch, windows_zipfile, execute, and other dynamic behavior require the not_if/only_if guards for idempotency
- reboots
  - enabling/disabling certain windows features during a chef run may require a mid-run reboot; ex: UAC

windows web server management
management lifecycle

- bootstrapping chef client
- creating the server
- deploying the web app
- updating the web app

windows web server management
management lifecycle

- bootstrapping chef client
- creating the server
- deploying the web app
- updating the web app

windows web server management
management lifecycle

- bootstrapping chef client
- creating the server
- deploying the web app
- updating the web app

windows web server management
management lifecycle

- bootstrapping chef client
- creating the server
- deploying the web app
- updating the web app

bootstrapping chef client

plan your applications workflow and navigation

bootstrapping
general overview

What exactly does the bootstrap do?

bootstrapping
general overview

- downloads and installs chef client
- configures the chef client
- downloads keys
  - validation
  - data bag
- runs chef client

bootstrapping
general overview

- downloads and installs chef client
- configures the chef client
- downloads keys
  - validation
  - data bag
- runs chef client

bootstrapping
general overview

- downloads and installs chef client
- configures the chef client
- downloads keys
  - validation
  - data bag
- runs chef client

bootstrapping
general overview

- downloads and installs chef client
- configures the chef client
- downloads keys
  - validation
  - data bag
- runs chef client

bootstrapping
general overview

- downloads and installs chef client
- configures the chef client
- downloads keys
  - validation
  - data bag
- runs chef client

bootstrapping
windows; not a standard bootstrap

- no *out of the box* ssh solution
- winrm is closest equivalent
  - pros
    - ability to run commands designed to run locally
    - ability to run powershell scripts
  - cons
    - disabled or not configured by default
    - creates additional openings for an attacker to exploit
    - requires a web server to be running on the machine

bootstrapping
windows; not a standard bootstrap

- no *out of the box* ssh solution
- winrm is closest equivalent
  - pros
    - ability to run commands designed to run locally
    - ability to run powershell scripts
  - cons
    - disabled or not configured by default
    - creates additional openings for an attacker to exploit
    - requires a web server to be running on the machine

bootstrapping
windows; not a standard bootstrap

- no *out of the box* ssh solution
- winrm is closest equivalent
  - pros
    - ability to run commands designed to run locally
    - ability to run powershell scripts
  - cons
    - disabled or not configured by default
    - creates additional openings for an attacker to exploit
    - requires a web server to be running on the machine

bootstrapping
windows; not a standard bootstrap

- no *out of the box* ssh solution
- winrm is closest equivalent
  - pros
    - ability to run commands designed to run locally
    - ability to run powershell scripts
  - cons
    - disabled or not configured by default
    - creates additional openings for an attacker to exploit
    - requires a web server to be running on the machine

bootstrapping
windows; not a standard bootstrap

- no *out of the box* ssh solution
- winrm is closest equivalent
  - pros
    - ability to run commands designed to run locally
    - ability to run powershell scripts
  - cons
    - disabled or not configured by default
    - creates additional openings for an attacker to exploit
    - requires a web server to be running on the machine

bootstrapping
windows; not a standard bootstrap

- no *out of the box* ssh solution
- winrm is closest equivalent
  - pros
    - ability to run commands designed to run locally
    - ability to run powershell scripts
  - cons
    - disabled or not configured by default
    - creates additional openings for an attacker to exploit
    - requires a web server to be running on the machine

bootstrapping
windows; not a standard bootstrap

- no *out of the box* ssh solution
- winrm is closest equivalent
  - pros
    - ability to run commands designed to run locally
    - ability to run powershell scripts
  - cons
    - disabled or not configured by default
    - creates additional openings for an attacker to exploit
    - requires a web server to be running on the machine

bootstrapping
your options

unmodified windows base image

- no remote bootstrap options available
- local bootstrap via manual install or custom script

modified windows base image

- bootstrap via the knife plugin
- bootstrap via a custom script on first start*

* some cloud services, like AWS, provide base images with the ability to run a custom script at first startup baked in

bootstrapping
your options

unmodified windows base image

- no remote bootstrap options available
- local bootstrap via manual install or custom script

modified windows base image

- bootstrap via the knife plugin
- bootstrap via a custom script on first start*

* some cloud services, like AWS, provide base images with the ability to run a custom script at first startup baked in

bootstrapping
modified image: knife plugin

- requires winrm to be installed and configured
- allows remote bootstrap of any accessible machine
- ability to force a chef run remotely

bootstrapping
modified image: knife plugin

- requires winrm to be installed and configured
- allows remote bootstrap of any accessible machine
- ability to force a chef run remotely

bootstrapping
modified image: knife plugin

- requires winrm to be installed and configured
- allows remote bootstrap of any accessible machine
- ability to force a chef run remotely

bootstrapping
modified image: custom script

custom powershell

- requires ability to run custom script on first start
- the custom script can download and run a bootstrap script hosted on another server
  - canned bootstrap scripts for different roles
  - dynamically generated bootstrap for each different role

bootstrapping
modified image: custom script

custom powershell

- requires ability to run custom script on first start
- the custom script can download and run a bootstrap script hosted on another server
  - canned bootstrap scripts for different roles
  - dynamically generated bootstrap for each different role

bootstrapping
modified image: custom script

custom powershell

- requires ability to run custom script on first start
- the custom script can download and run a bootstrap script hosted on another server
    - canned bootstrap scripts for different roles
    - dynamically generated bootstrap for each different role

bootstrapping
picking your modified image method

Questions you should ask first

- winrm
  - are your servers on the public internet?
    - winrm can increase your attack vector if not locked down properly
  - are you already going to run a web server?
  - are you going to be running the chef-client as a service?
- custom script
  - do you have an a place to host the canned or dynamic scripts?
  - do you need to execute anything during the bootstrap that’s not easily templatable?

requires	winrm	firewall changes	knife plugin	external script host	powershell
knife	X	X	X		X
custom script				X	X

bootstrapping with knife

bootstrapping the client

bootstrapping
knife: base image prep

- install/configure winrm

```
$ winrm quickconfig -q
$ winrm set winrm/config/winrs @{MaxMemoryPerShellMB="300"}
$ winrm set winrm/config @{MaxTimeoutms="1800000"}
$ winrm set winrm/config/service @{AllowUnencrypted="true"}
$ winrm set winrm/config/service/auth @{Basic="true"}
```

- update firewall

```
$ netsh advfirewall firewall set rule name="Windows Remote Management (HTTP-In)"
profile=public protocol=tcp localport=5985 remoteip=localsubnet new remoteip=any
```

source: ops code docs

bootstrapping
knife: workstation prep

- install the knife-windows gem

```
$ gem install knife-windows
```

- update/modify the default template
    - windows-chef-client-msi.erb
    - add any custom bootstrap work
- run knife to bootstrap the client machine

```
$ knife bootstrap windows winrm your.machine.com -r 'role[foo]'
-x Administrator -P 'password'
```

bootstrapping with a custom script

bootstrapping the client

bootstrapping
what the powershell bootstrap does

- downloads and installs the chef client
- downloads the data bag encryption key *
  - if you store anything in an encrypted data bag that is needed during the chef run
- downloads the validation key *
- creates the chef client configuration file
- runs the chef client with the specified role(s)

* storing your keys on the same server as your code defeats the purpose

bootstrapping
what the powershell bootstrap does

- downloads and installs the chef client
- downloads the data bag encryption key *
  - if you store anything in an encrypted data bag that is needed during the chef run
- downloads the validation key *
- creates the chef client configuration file
- runs the chef client with the specified role(s)

* storing your keys on the same server as your code defeats the purpose

bootstrapping
what the powershell bootstrap does

- downloads and installs the chef client
- downloads the data bag encryption key *
  - if you store anything in an encrypted data bag that is needed during the chef run
- downloads the validation key *
- creates the chef client configuration file
- runs the chef client with the specified role(s)

* storing your keys on the same server as your code defeats the purpose

bootstrapping
what the powershell bootstrap does

- downloads and installs the chef client
- downloads the data bag encryption key *
  - if you store anything in an encrypted data bag that is needed during the chef run
- downloads the validation key *
- creates the chef client configuration file
- runs the chef client with the specified role(s)

* storing your keys on the same server as your code defeats the purpose

bootstrapping
what the powershell bootstrap does

- downloads and installs the chef client
- downloads the data bag encryption key *
  - if you store anything in an encrypted data bag that is needed during the chef run
- downloads the validation key *
- creates the chef client configuration file
- runs the chef client with the specified role(s)

* storing your keys on the same server as your code defeats the purpose

bootstrapping
bootstrapping the bootstrap

download and run your bootstrap using a powershell script

```
$ @powershell -NoProfile -ExecutionPolicy unrestricted -Command "iex ((new-object
net.webclient).DownloadString('https://your.server.com/your-bootstrap'))"
```

if your server uses a self-signed SSL cert add the following before invoking the expression (iex)

```
[Net.ServicePointManager]::ServerCertificateValidationCallback = {$true}
```

bootstrapping
powershell: downloading files

```
Function DownloadFileFromWeb($url, $file)
{
    [Net.ServicePointManager]::ServerCertificateValidationCallback = {$true}
     Write-Host "Downloading $file..."
     if(-not (Test-Path $file))
     {
          (New-Object System.Net.WebClient).DownloadFile($url,$file)
     }
}
```

bootstrapping
powershell: installing the chef client

```
Function InstallChefClient($installer)
{
    Write-Host "Installing Chef Client..."
    Start-Process -FilePath msiexec -ArgumentList /i, $installer, '/L c:\chef\log
     \install.log', /quiet -Wait
}
```

bootstrapping

powershell: creating a client config

```
Function CreateClientFile($chefDirectory)
{
	Write-Host "Creating $chefDirectory/client.rb..."
	$clientRB = @"
node_name                  "YOUR_NODE_NAME"`r`n
log_level                  :info`r`n
verbose_logging            true`r`n
log_location               "c:/chef/log/chef-client.log"`r`n
file_cache_path            "c:/chef/cache"`r`n
file_backup_path           "c:/chef/backup"`r`n
cache_options              ({:path => "c:/chef/cache/checksums", :skip_expires => true})`r`n
chef_server_url            "https://api.opscode.com/organizations/YOUR_ORG"`r`n
validation_client_name     "validator"`r`n
validation_key             "c:/chef/validator.pem"`r`n
client_key                 "c:/chef/client.pem"`r`n
data_bag_decrypt_minimum_version 2`r`n
"@
	$file = "$chefDirectory/client.rb"
	if((Test-Path $file))
	{
		Clear-Content "$file"
	}
	Add-Content "$chefDirectory/client.rb" "$clientRB"
}
```

bootstrapping
powershell: running the chef client

```
Function RunChefClient
{
        Write-Host "Running chef client for the first time..."
        Start-Process -FilePath C:/opscode/chef/bin/chef-client -ArgumentList "-o role[SERVER_ROLE]
        -E ENVIRONMENT_FILE" -Wait
}
```

bootstrapping
powershell: execute the bootstrap

```
$server = ’https://your.server’

DownloadFileFromWeb "https://www.opscode.com/chef/install.msi" "c:/chef/chef-client-install.msi"
InstallChefClient("c:\chef\chef-client-install.msi")
DownloadFileFromWeb "$server/encrypted_data_bag_key.txt" "c:/encrypted_data_bag_key"
DownloadFileFromWeb "$server/validator.pem" "c:/chef/validator.pem"
CreateClientFile ”c:/chef/”
RunChefClient
```

creating the server

setup the server and install your app

creating the server
step by step

- install iis
- install .net
- install web deploy

creating the server
install iis

- add iis features to attribute file
- create recipe to install features

install iis
install features: attributes

- your_cookbook/attributes/default.rb

```
default['iis']['features_list'] = ["IIS-WebServerRole"]
default['iis']['features_list'] << "IIS-WebServer"
default['iis']['features_list'] << "IIS-CommonHttpFeatures"
default['iis']['features_list'] << "IIS-HttpRedirect"
default['iis']['features_list'] << "IIS-ISAPIFilter"
default['iis']['features_list'] << "IIS-ISAPIExtensions"
default['iis']['features_list'] << "IIS-NetFxExtensibility"
default['iis']['features_list'] << "IIS-ASPNET"
default['iis']['features_list'] << "IIS-HostableWebCore"
default['iis']['features_list'] << "IIS-WindowsAuthentication"
default['iis']['features_list'] << "NetFx3"
default['iis']['features_list'] << "MicrosoftWindowsPowerShellISE"
default['iis']['features_list'] << "WAS-WindowsActivationService"
default['iis']['features_list'] << "WAS-ConfigurationAPI"
default['iis']['features_list'] << "WAS-NetFxEnvironment"
```

creating the server
install iis

- add iis features to attribute file
- create recipe to install features

install iis
install features: recipe

- your_cookbook/recipes/iis_features.rb

```
node['iis']['features_list'].each do |feature|
     windows_feature feature do
          action :install
     end
end
```

creating the server
step by step

- install iis
- install .net
- install web deploy

creating the server
install .net

- register .net 4 with iis
- install .net 4.5

install .net

register .net 4 with iis

- your_cookbook/recipes/register_aspnet.rb

```
fx_path = "C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319"
regiis_exe = "aspnet_regiis.exe"

execute "Register ASP.NET MVC" do
    command "#{fx_path}\\#{regiis_exe} -iru"
    action :run
end
```

creating the server
install .net

- register .net 4 with iis
- install .net 4.5

install .net

install .net 4.5

- your_cookbook/attributes/default.rb

```
default['package_sources']['dotnet_4_5'] = "http://..."
```

- your_cookbook/recipes/install_dotnet_4_5.rb

```
windows_package "Microsoft .NET Framework 4.5" do
    source node.default['package_sources']['dotnet_4_5']
    action :install
    options "/q"
    timeout 1200
    installer_type :inno
end
```

creating the server
step by step

- install iis
- install .net
- install web deploy

install .net
install web deploy 2

- your_cookbook/attributes/default.rb

```
default['package_sources']['web_deploy'] = "http://..."
```

- your_cookbook/recipes/install_webdeploy.rb

```
windows_package "Microsoft Web Deploy 2.0" do
    source node.default['package_sources']['web_deploy']
    action :install
    options "/qn /norestart"
    installer_type :msi
end
```

deploying the web app

getting your app deployed

creating the server
setup your app

- create firewall lwrp
- create web_deploy lwrp
- create recipe which will deploy the app
    - create iis app pool
    - create iis site
    - open firewall for app
    - deploy app
    - start app pool

deploying the web app

firewall lwrp: resource

```
actions :add

attribute :rule_name, :kind_of => String, :name_attribute => true
attribute :firewall_action, :kind_of => Symbol, :default => :Allow, :equal_to => [:Allow, :Block]
attribute :direction, :kind_of => Symbol, :default => :In, :equal_to => [:In, :Out]
attribute :protocol, :kind_of => Symbol, :default => :tcp, :equal_to => [:tcp, :udp, :icmpv4, :icmpv6, :any]
attribute :ports, :kind_of => Array, :default => [80]

attr_accessor :created


def initialize(name, run_context=nil)

        super

        @action = :add

end
```

deploying the web app
firewall lwrp: provider

```
require 'chef/mixin/shell_out'
include Chef::Mixin::ShellOut

action :add do
	if @new_resource.created
		Chef::Log.info "#{@new_resource.rule_name} is already created"
	else
		cmd = "netsh advfirewall firewall add rule"
		cmd << " Name=\"#{@new_resource.rule_name}\""
		cmd << " Dir=\"#{@new_resource.direction.to_s}\""
		cmd << " Action=\"#{@new_resource.firewall_action.to_s}\""
		cmd << " Protocol=\"#{@new_resource.protocol.to_s}\""
		cmd << " Localport="
		cmd << @new_resource.ports.join(",")

		Chef::Log.debug(cmd)
		shell_out!(cmd)
	end
end
```

deploying the web app
firewall lwrp: idempotence

```
def load_current_resource
        cmd_base = "netsh advfirewall firewall show rule"
        cmd_name = "Name=\"#{@new_resource.rule_name}\""
        cmd = shell_out("#{cmd_base} #{cmd_name}", { :returns => [0] })
        if (cmd.stderr.empty? && (cmd.stdout =~ /^.*Rule Name.*$/i))
          @new_resource.created = true
        end
end
```

creating the server
setup your app

- create firewall lwrp
- create web_deploy lwrp
- create recipe which will deploy the app
  - create iis app pool
  - create iis site
  - open firewall for app
  - deploy app
  - start app pool

deploying the web app
web_deploy lwrp: resource

```
actions :sync

attribute :package, :kind_of => String, :name_attribute => true
attribute :destination, :kind_of => String, :default => “auto”
attribute :parameters, :kind_of => String

def initialize(name, run_context=nil)
        super
        @action = :sync
end
```

deploying the web app
web_deploy lwrp: provider

```
action :sync do
    msdeploy_cmd = "\"%programfiles%\\IIS\\Microsoft Web Deploy V2\
    \msdeploy.exe\" "
    msdeploy_cmd << "-verb:sync "
    msdeploy_cmd << "-source:package=\"#{@new_resource.package}\" "
    msdeploy_cmd << "-dest=\"#{@new_resource.destination}\" "

    @new_resource.parameters.each do |name, value|
        msdeploy_cmd << "-setParam:name=\"#{name}\",value=\"#{value}\" "
    end unless @new_resource.parameters.nil?

    execute "webdeploy" do
        command msdeploy_cmd
    end
end
```

creating the server
setup your app

- create firewall lwrp
- create web_deploy lwrp
- create recipe which will deploy the app
    - create iis app pool
    - create iis site
    - open firewall for app
    - deploy app
    - start app pool

deploying the web app
create iis app pool

```
# create iis app pool
iis_pool node['webapp']['name'] do
        runtime_version node['webapp']['app_pool']['runt...ion']
        pipeline_mode :Integrated
        action [:add, :config, :stop]
end
```

creating the server
setup your app

- create firewall lwrp
- create web_deploy lwrp
- create recipe which will deploy the app
  - create iis app pool
  - create iis site
  - open firewall for app
  - deploy app
  - start app pool

deploying the web app

create iis site

```
# create iis site directory
directory "#{node['iis']['docroot']}/#{node['webapp']['name']}" do
        recursive true
        action :create
end


# create iis site
iis_site "#{node['webapp']['name']}" do
        site_name "#{node['webapp']['name']}"
        port node['webapp']['site']['config']['port']
        path "#{node['iis']['docroot']}/#{node['webapp']['name']}"
        application_pool node['webapp']['name']
        action [:add,:start]
end
```

creating the server
setup your app

- create firewall lwrp
- create web_deploy lwrp
- create recipe which will deploy the app
  - create iis app pool
  - create iis site
  - open firewall for app
  - deploy app
  - start app pool

deploying the web app
open firewall for app

```
# open firewall for app
your_cookbook_firewall node['firewall']['rule_name'] do
        action :add
        firewall_action :Allow
        direction :In
        protocol :tcp
        ports node['firewall']['ports']
end
```

creating the server
setup your app

- create firewall lwrp
- create web_deploy lwrp
- create recipe which will deploy the app
    - create iis app pool
    - create iis site
    - open firewall for app
    - deploy app
    - start app pool

deploying the web app

deploy app

```
# create temp directory
directory "#{node['webapp']['local_directory]}" do
          recursive true
          action :create
end

# copy web app to temp directory
cookbook_file node['webapp']['package'] do
          path "#{node['webapp']['local_directory']}/#{node['webapp']['package']}"
          action :create
end

# deploy app
your_cookbook_web_deploy "#{node['webapp']['local_directory']}/
    #{node['webapp']['package']}" do
          parameters node['webapp']['parameters']
end
```

creating the server
setup your app

- create firewall lwrp
- create web_deploy lwrp
- create recipe which will deploy the app
  - create iis app pool
  - create iis site
  - open firewall for app
  - deploy app
  - start app pool

deploying the web app

start app pool

```
# start app pool
iis_pool node['webapp']['name'] do
        action :start
end
```

simplifying your setup

using some chef built-in goodness

simplifying your setup
cookbook default recipes

- your_cookbook/recipes/default.rb

```
include_recipe "your_cookbook::iis_features"
include_recipe "iis::remove_default_site"
include_recipe "your_cookbook::install_dotnet_4_5"
include_recipe "your_cookbook::register_aspnet"
include_recipe "your_cookbook::install_webdeploy"
include_recipe "your_cookbook::deploy_web_app"
```

- add recipe[your_cookbook] in your role’s run list

simplifying your setup
cookbook default recipes

- your_cookbook/recipes/default.rb

```
include_recipe "your_cookbook::iis_features"
include_recipe "iis::remove_default_site"
include_recipe "your_cookbook::install_dotnet_4_5"
include_recipe "your_cookbook::register_aspnet"
include_recipe "your_cookbook::install_webdeploy"
include_recipe "your_cookbook::deploy_web_app"
```

- add recipe[your_cookbook] in your role’s run list

simplifying your setup
cookbook role

- create a role

```
$ knife role create MyWebApp
```

- Add your role information

```
{
    "name": "MyWebApp",
    "description": "Windows Server With Our Web App",
    "json_class": "Chef::Role",
    "chef_type": "role",
    "run_list": [
        "recipe[your_cookbook]"
    ]
}
```

simplifying your setup
cookbook role

- create a role

```
$ knife role create MyWebApp
```

- Add your role information

```
{
    "name": "MyWebApp",
    "description": "Windows Server With Our Web App",
    "json_class": "Chef::Role",
    "chef_type": "role",
    "run_list": [
        "recipe[your_cookbook]"
    ]
}
```

where to go from here

managing your windows web server post setup

where to go from here
deployment

- re-deploying code will cause app pool reset
  - options
    - plan rolling deployments into your deployment process
    - use blue-green deployments

questions?

office hours
4:15pm - 4:35pm
marina room

Paul Oremland
http://paul.oremland.net
http://tech.infospace.com
https://github.com/poremland

Resources

- paul oremland’s github:
  https://github.com/poremland
- paul oremland’s blog:
  http://paul.oremland.net
- infospace technology blog:
  http://tech.infospace.com/
- blue-green deployments:
  http://martinfowler.com/bliki/BlueGreenDeployment.html
- winrm:
  http://msdn.microsoft.com/en-us/library/aa384372(v=vs.85).aspx