Yushu Yao 姚昱書, Lawrence Berkeley National Lab

Trust VM Creation Recipes, not Images

A scheme for defining and deploying trusted Virtual Machines to Grid Sites

Start Here

The Fight - VOs vs. Sites

✦ VO says to Site: I must customize the VM image that you run. I need to configure it to fit my needs, I need to install my application stack. So I can run my software.

✦ Site says to VO: I don't TRUST you!!! I will not run an image you provide, because no matter what you do, according to my 100-page long security policies, it is insecure.

Problem

✦ To fully leverage the benefits of Virtualization, VOs ***need*** to decide what they run in the VM. However, sites have security policies to follow.

✦ There has to be a way to let Sites trust the VMs that VO provide.

✦ Current studies concentrate on VM Images (e.g. to create, distribute, authorize and revoke them)

Solution - Recipes

The more efficient alternative it to work with the ***Recipes*** to Create VM images, instead of the **Images** themselves. In this way:

✦ ***VOs*** will have the freedom to customize the VM they want to run.

✦ ***Sites*** can run **VO-provided** VM without sacrificing security. (E.g. not giving out root access on any running instance)

Two Sets of Recipes

✦ **What is a Recipe?**

✦ A set of scripts that can install applications and configure a linux installation (Usually **needs root** privilege to execute)

✦ **Two Sets of Recipes**:

✦ **VO Recipe**: defines the software/services the VO need to run. It will run as **root** at deploy time at Sites.

✦ **Site Recipe**: defines the security policies of the Site and attach Site specific resources.

Key Ideas

✦ **Building the VM:**

✦ Deployment start with the baseline image

✦ Install/configure software only at deploy time, instead of pre-configured images.

✦ Both VO and Site Recipes are executed with root privilege. Thus giving VO maximum flexibility.

✦ **Security**

✦ VO Recipes can be examined for venerability.

✦ Apply Site Recipe after VO Recipe to ensure Site Security Policies.

✦ Site needs not give out root privilege on any running instance.

Why Recipes, Why Not Images

✦ ***Images*** are black boxes, normal linux has >50k files, impossible to fully exam for vulnerabilities, even automatically.

✦ ***Recipes*** are in text format, much easier to exam.

✦ ***Images*** are huge (GB), they are hard to transfer, storage and even zipping, hashing.

✦ ***Recipes*** are much smaller (KB), so every operation is faster.

✦ Versioning Control of binary **Images** is a nightmare. Versioning Control of **Recipes** is much easier (Simply put it into SVN). Making project management much easier.

✦ **Recipes** are highly **reusable** - one VO Recipe can work for many Sites, one Site Recipe can work for many VOs.

Puppet - The Recipe Language

✦ **Requirements for the Recipe Language:**

✦ VOs/Sites are not always willing to learn new technologies: **Easy to Use**

✦ It must define complex software security configuration details: **Powerful and Reliable**

✦ Recipes are not applied until deploy time: **Fast Executing**

✦ Many VMs might be deployed at the same time: **Scalable**

✦ **Best Candidate: Puppet (http://www.puppetlabs.com/)**

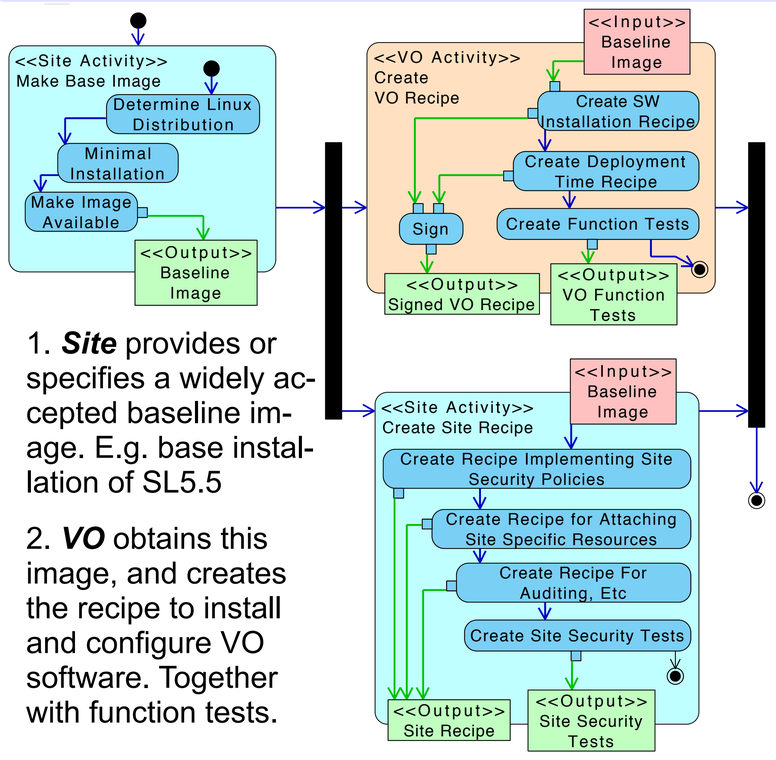
✦ Wildly adopted (Sun, RedHat, Twitter are using it)

✦ Ruby-like syntax, fast, scalable and powerful

✦ Many Site admins are already using it.

Below are the UML diagrams showing each stage in the VM's Lifecycle. → Object Flow → Control Flow

1. Create

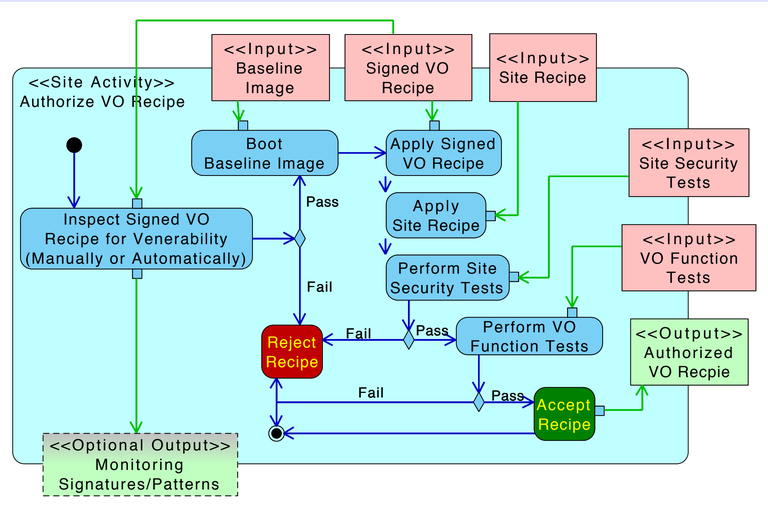


1. ***Site*** provides or specifies a widely accepted baseline image. E.g. base installation of SL5.5

2. ***VO*** obtains this image, and creates the recipe to install and configure VO software. Together with function tests.

3. ***Site*** creates site recipe applying security policies, create security tests, attache site specific resources (e.g. mount GPFS)

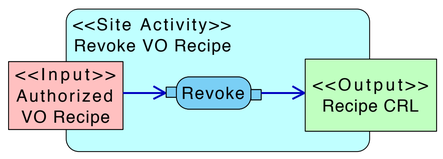
2. Authorize



It is the ***Site***'s responsibility to authorize a VO Recipe.

1. ***Site*** examines the VO Recipe for obvious defects. (It's much easier to exam a script than an image file)
2. ***Site*** boots the Baseline Image, apply VO and Site recipe, and perform the functional an security tests
3. If all tests are passed, sign the VO's recipe (like signing a X509 certificate)

3. Revoke



Sites can revoke an Authorized VO Recipe.

- Similar to X509 CRL mechanism, a Recipe CRL is used to keep track of revoked Recipes.
- At deploy time, Site checks Recipe CRL to make sure a VO recipe is not revoked.

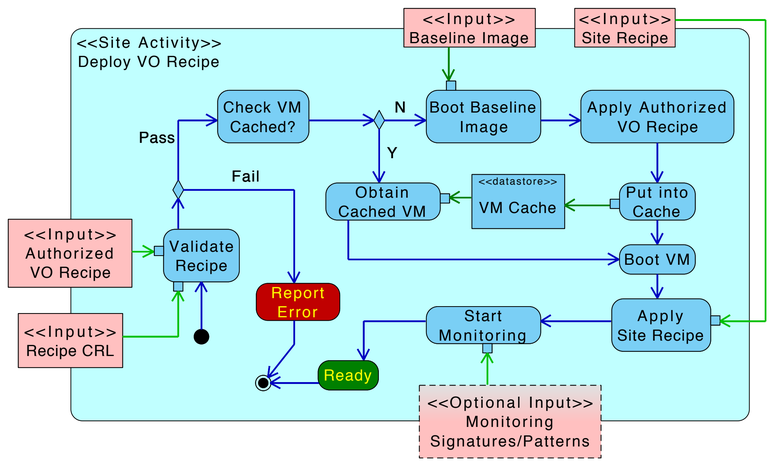
4. Catalog & Version Control

Cataloging becomes very simple since we are dealing with Recipes (Scripts) instead of binary images.

5. Transfer

Transferring Recipes (KiloBytes) instead of images (GigaBytes). Huge savings.

6. Deploy



To deploy an Authorized VO Recipe:

1. First check to see if the recipe has been revoked.
2. If not, it will start up the baseline image, apply VO recipes then Site recipes. (Site recipes are applied later to close any "backdoors" VO recipe left open).
3. Since applying VO recipe might take some time, the result of this step can be cached for faster deployment in the future.
4. Optionally, the site can start monitoring the running instance for security or performance oddities.

Wanted: Pioneer Sites and VOs

We need VOs and Sites to work together !
Already one VO showed interest.
Questions or Comments to: Yushu Yao, yao.yushu@gmail.com

Yushu Yao 姚昱書, Lawrence Berkeley National Laboratory

Special Thanks to: **Ian Hinchliffe, Paolo Calafiura, Charles Leggett**

CHEP 2010 - October 18-22 2010