cohesive networks

VNS3:turret
WAF Guide
Sept 2015

Table of Contents

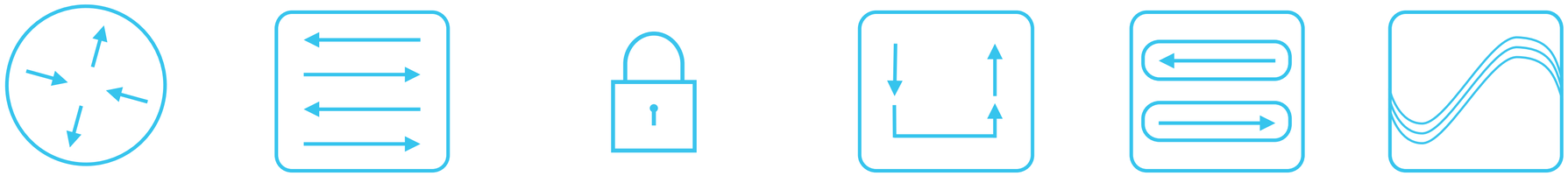
Introduction

VNS3:turret provides container based network services

Isolated Linux containers within VNS3 allows Partners and Customers to embed features and functions safely and securely into their Cloud Network.

Proxy | Reverse Proxy | Content Caching | Load Balancer | IDS | Custom Container

VNS3 Core Components



Router | Switch | Firewall | Protocol Redistributor | VPN Concentrator | Scriptable SDN

Getting Help with VNS3

This document assumes you have a VNS3 Controller instance launched and running in a security group, network or similar that has the appropriate access rules included for normal VNS3 operations.

See the specific instructions for your cloud setup and instance launch on our Product Resources page.

Please review the VNS3 Support Plans and Contacts before sending support inquiries.

Requirements

You have a cloud or virtual infrastructure account that Cohesive Networks can use for enabling your access to the VNS3 Controller Images.

Ability to configure a client (whether desktop based or cloud based) to use OpenVPN client software.

You have agreed to the VNS3 Terms and Conditions.

Basic knowledge of Linux software installation and use of command line tools.

Configurable Default WAF Plugin

VNS3:turret WAF overview

The VNS3:Turret system uses ModSecurity within Nginx as a web application firewall (WAF). This combination was chosen due to simplicity, high performance and proven durability and scale in large deployments such as CloudFlare.

The Nginx:ModSecurity combination is deployed to VNS3:turret using the containers mechanism. These instructions cover customisation of the container image that will be used so that customer keys and rule sets can be employed.

Please be familiar with the VNS3 plug-in configuration guide: http://cohesive.net/dnld/Cohesive-Networks_VNS3-3.5-Container-System.pdf

Getting the Default WAF Plug-In

The Linux Container default plug in is accessible at the following URL:

https://vns3-containers-read-all.s3.amazonaws.com/WAF_modsecurity_Base/WAF_modsecurity_Base.export.tar.gz

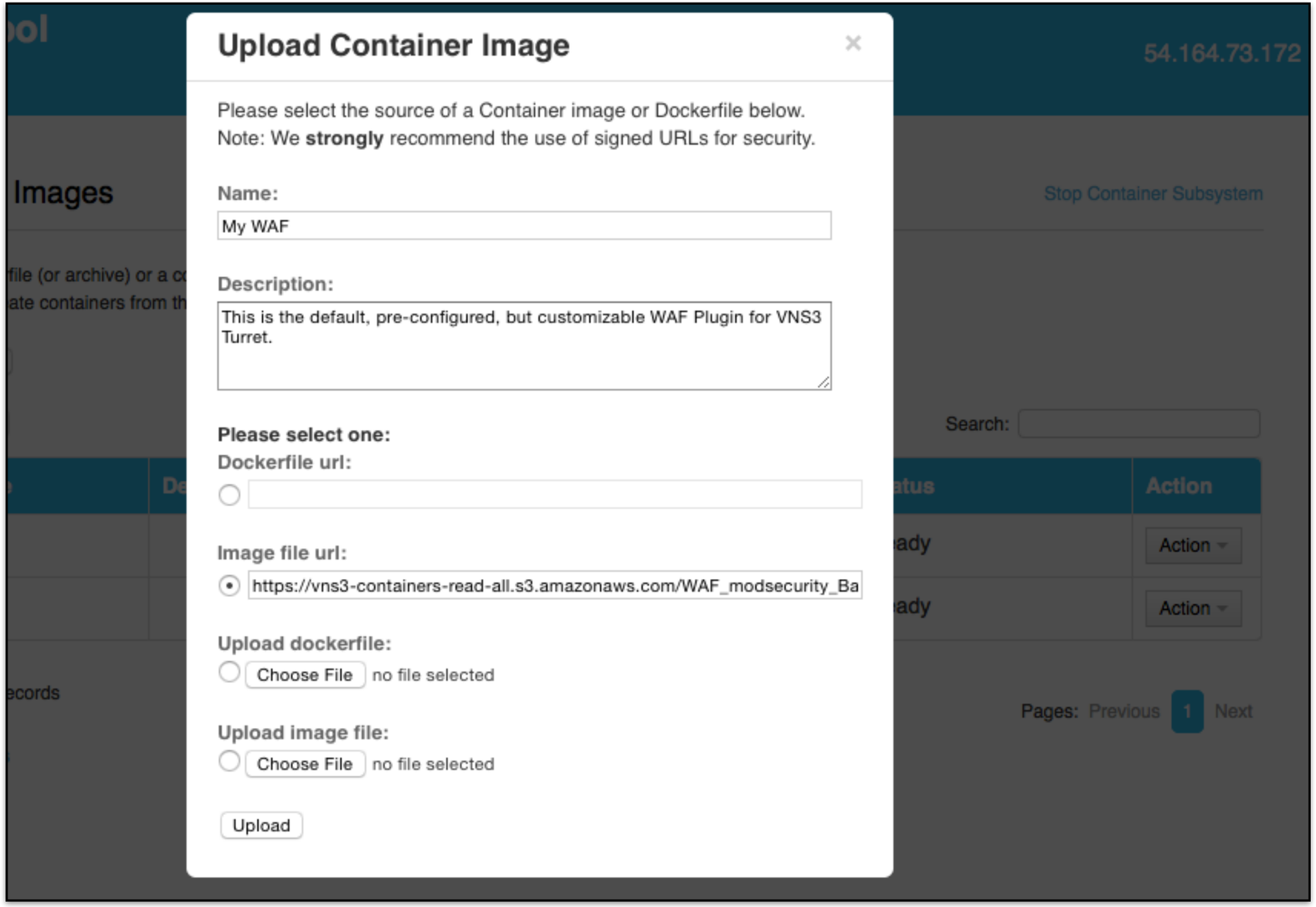
This is a read only Amazon S3 storage location. Only Cohesive Networks can update or modify files stored in this location.

This URL can be used directly in a VNS3 Controller via the Web UI or API to import the container for use into that controller. (General screenshot walkthrough and help available in the plug-in configuration document.)

Getting the Default WAF Plug-In

From the “Container —> Images” menu item, choose “Upload Image”.

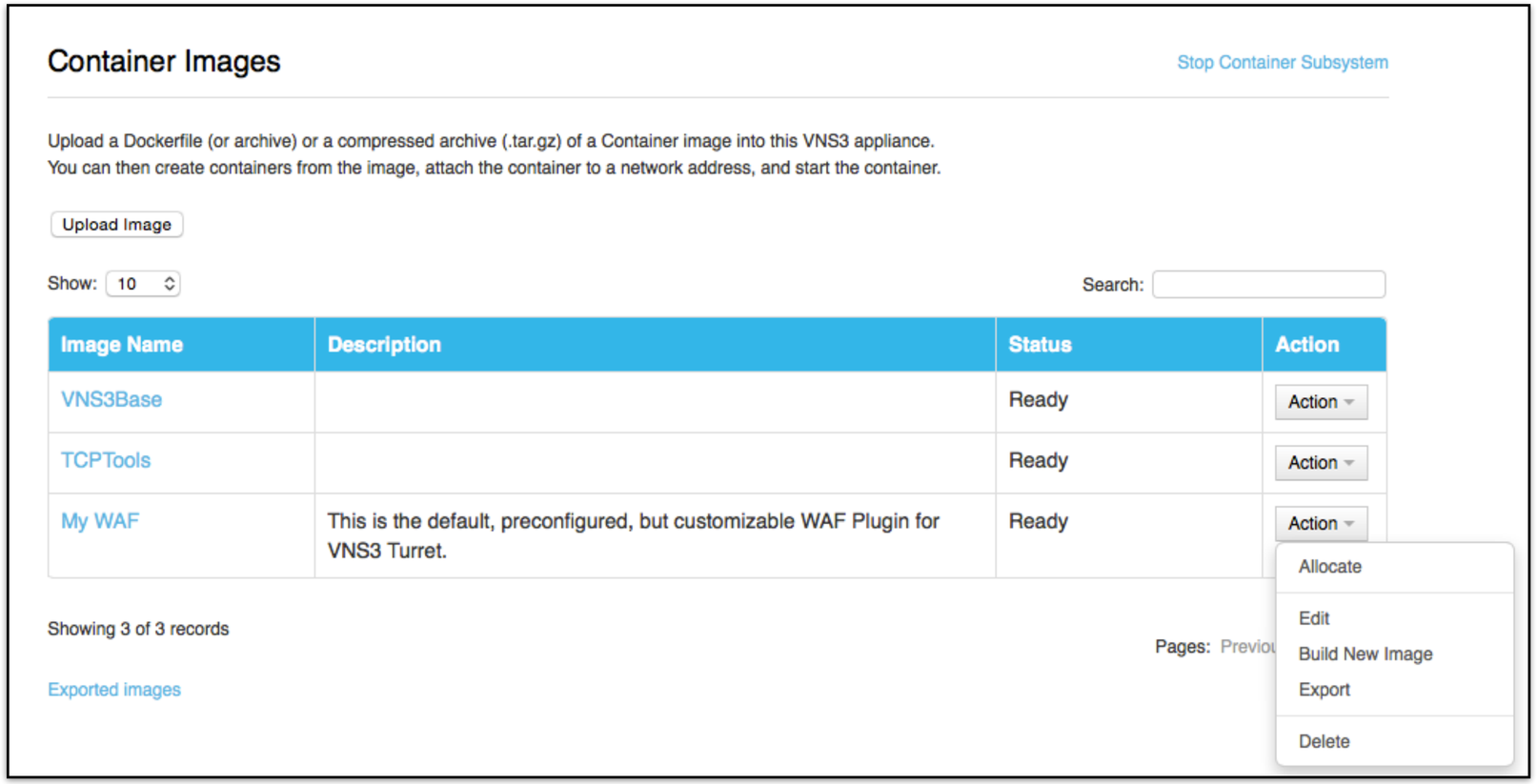
To use the pre-configured plugin paste the URL into the “Image File URL” box.



Getting the Default WAF Plug-In

When the Image has imported it will say “Ready” in the Status Column.

To then launch a running WAF container, choose “Allocate” from the “Action” menu.



Launching a WAF Container

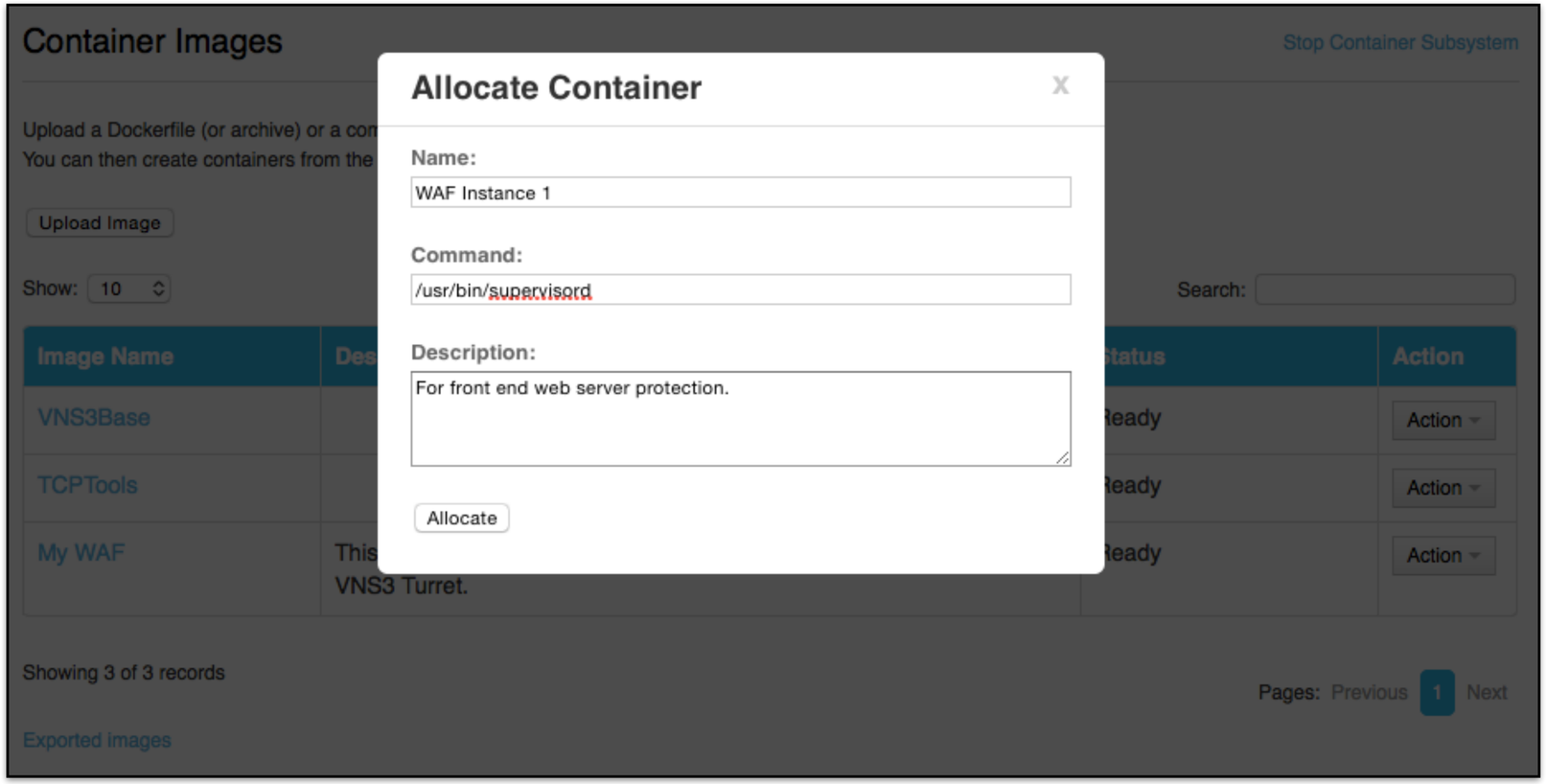
After selecting “Allocate” from the “Actions” menu you then name your container, provide a description and the command used to execute the container.

The name and description should be something meaningful within the context of your organization and its policies.

In MOST cases the command used to run plugin containers will be: /usr/bin/supervisord

However, this may vary with individual containers, please consult each plug-in’s specific documentation.

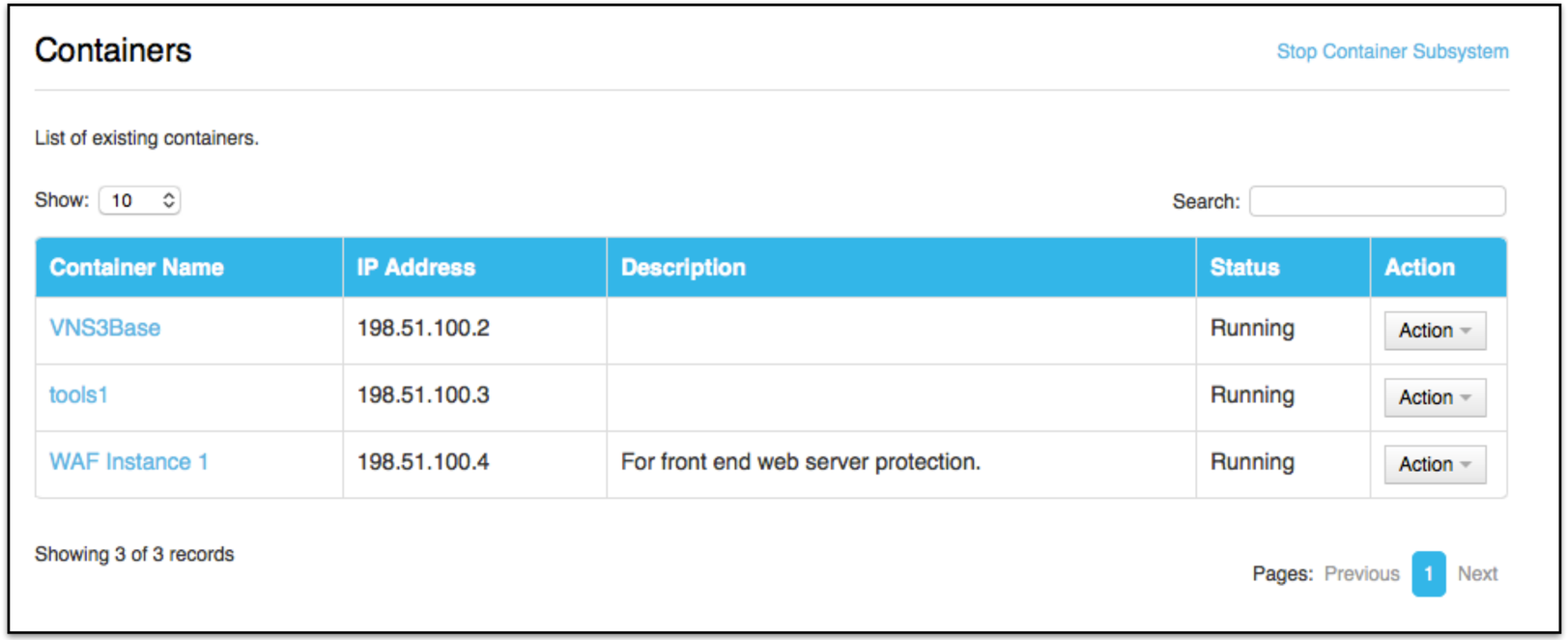
The command to run the WAF container is: /usr/bin/supervisord



Confirming the WAF Container is running

After executing the “Allocate” operation you will be taken to the Container Display page.

You should see your WAF Container with the name you specified. The Status should be “Running” and it should have been given an IP address on your internal plug-in subnet (in this case 192.51.100.4).



Customizing Default WAF Plugin

Accessing the WAF Container

Accessing a Container from the Public Internet or your internal subnets will require additions to the inbound hypervisor firewall rules with the VNS3 Controller as well as VNS3 Firewall.

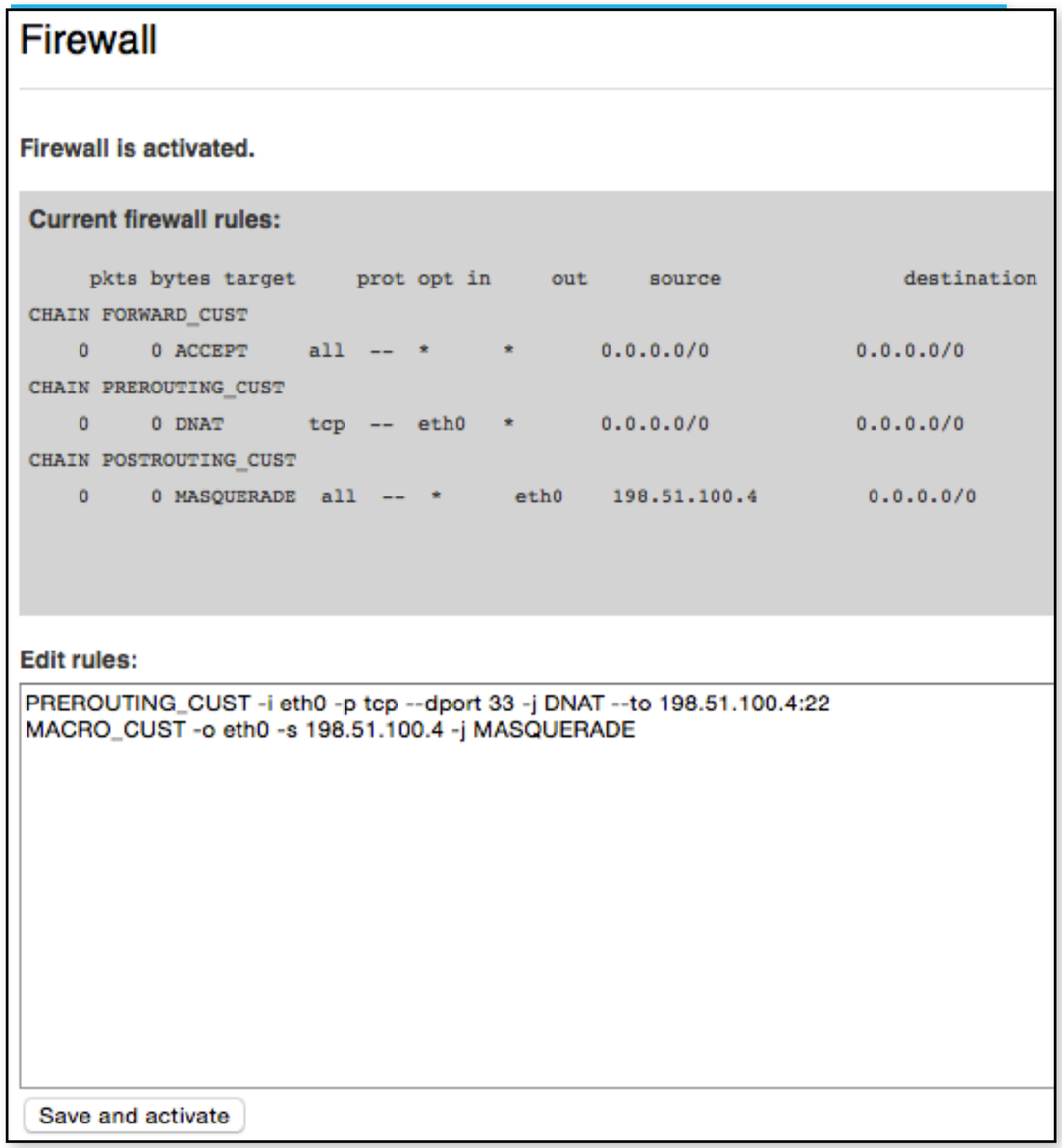
The following example shows how to access an SSH server running as a Container listening on port 22.

Network Firewall/Security Group Rule
Allow port 22 from your source IP or subnets.

VNS3 Firewall
Enter rules to port forward incoming traffic to the Container Network and Masquerade outgoing traffic off the VNS3 Controller's outer network interface.

#Let the Container Subnet Access the Internet Via the VNS3 Controller's Outer or Public IP
MACRO_CUST -o eth0 -s <WAF Container Network IP> -j MASQUERADE

#Port forward port 33 to the WAF Container port 22
PREROUTING_CUST -i eth0 -p tcp -s 0.0.0.0/0 --dport 33 -j DNAT --to <WAF Container Network IP>:22



Securing the WAF container

By default the WAF container has the following accounts, configured as described.

“root” - The root account is locked. The root account is not allowed to remote shell into the container. This is our recommended approach. However, if you wish to, you can use the “container_admin” account to unlock root, provide a root password, and edit /etc/ssh/sshd_config to allow remote login by root.

“container_admin” - The default password is “container_admin_123!”
The default demo public key is also installed in the /home/container_admin/.ssh/authorized_keys. **PLEASE change this password and this key** when configuring, or create a new default WAF image as your base for future use, following your authentication procedures. The account “container_admin” has “sudo” or superuser privileges, and is allowed to remote shell into the container.

Primary files for customization

There are two significant files for securing the WAF container:

/etc/ssh/sshd_config
Please ensure this file is configured to your organization's best practices.

/home/container_admin/.ssh/authorized_keys
The base container comes with an example public key installed, and private key for use in VNS3 documentation. Please remove after initial use or programmatic configuration.

Primary files for customization

The following files are the major elements for customizing the WAF container. It is not within the scope of this document to cover all elements of the included Nginx server, nor the Apache mod_security plugin, nor the OWASP rules set.

/home/container_admin/.ssh/authorized_keys (already discussed)

/etc/nginx/ssl/ssl.key
/etc/nginx/ssl/ssl.crt
Please replace with your own certs either self-generated or from a cert provider. These are default certs provided by Cohesive Networks for demonstration purposes.

Primary files for customization

/etc/nginx/nginx.conf
This file determines critical elements on how your WAF will respond at a base level to HTTP and HTTPS protocols. If you want to disallow use of SSLv3 for example, you would change this configuration. It also determines the listening port, and importantly the fact that "modsecurity" is in use.

The nginx.conf also determines whether to pass the traffic forward using port 80 or 443. Since the WAF Container is the first place the Web Application traffic will hit, it can receive on port 443 and then terminate SSL, passing the traffic on in plain HTTP (port 80). This is secure if the traffic is going to the Overlay which is encrypted already. If the traffic is going to be put into a Cloud or vSphere underlay, one might consider keeping it in HTTPS format.

The setting for "upstream appserver" determines the port used to forward the traffic on to the proxy. In the default configuration it is set to "server localhost:80;"

Primary files for customization

/etc/nginx/modsecurity.conf
Please read about the “modsecurity” capabilities and configuration here:
https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#Configuration_Directives

Key configuration items are:

SecRuleEngine - in preconfigured plugin set to “DetectionOnly” (vs. On or Off) means that the WAF will log/alert but will not block web traffic.

SecAuditLog - default log location is /var/log/modsec_audit.log, without configuring to send to a syslog this is where alerts can be seen.

SecStatusEngine - in preconfigured plugin is set to “Off”. If set to “On” it sends anonymized statistics to the ModSecurity project.

Primary files for customization

/etc/haproxy/haproxy.cfg
This file lets you specify the pool of web servers that the WAF elements are protecting and what their addresses are for forwarding the traffic.

Substitute your addresses for the ones in the default file saying:
“server app01 172.31.1.1:4567 cookie LSW_APP01 check"

If the Nginx server is sending the traffic on port 80 (as configured by default) the haproxy.cfg should have the entry “listen webcluster *:80”

Note, the HAProxy element has a statistic page available at the URL of the HAproxy server with “/stats” in the URL. There is a simple default username/password for the statistics page configured as “stats auth us3r:pa55Word”. Please disable stats by removing this line, or change the username and password.

The HAproxy element is running inside the container, and unless you do something via the VNS3 firewall the only access to it will be the Nginx server running in the same container, receiving traffic from the VNS3 firewall.

Primary files for customization

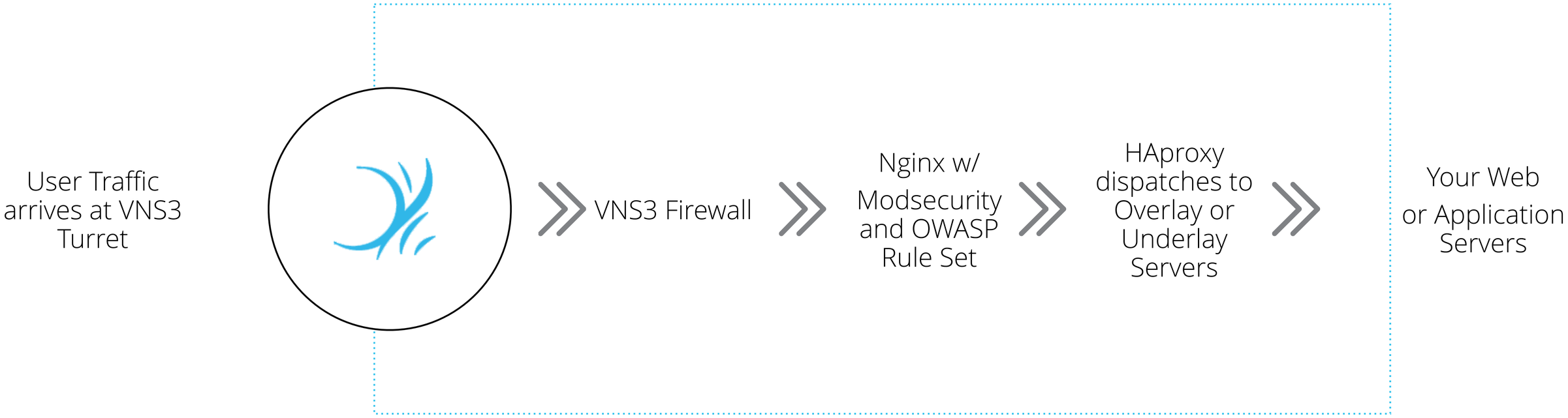
/etc/supervisor/conf.d/supervisord.conf

This file defines what services are started when the container is started. Looking at the default you will see Nginx, HAproxy, SSH, rsyslog.

Note: The rsylog component can be configured to copy information logged by the WAF to an external syslog server.

Putting it all together -
Getting traffic to your web servers via the VNS3 WAF Plugin

WAF Container Flow



The Web Application traffic arrives at VNS3 Turret where it is audited by the WAF components and then forwarded to the “true” web servers.

Forwarding Web Traffic to the WAF Container

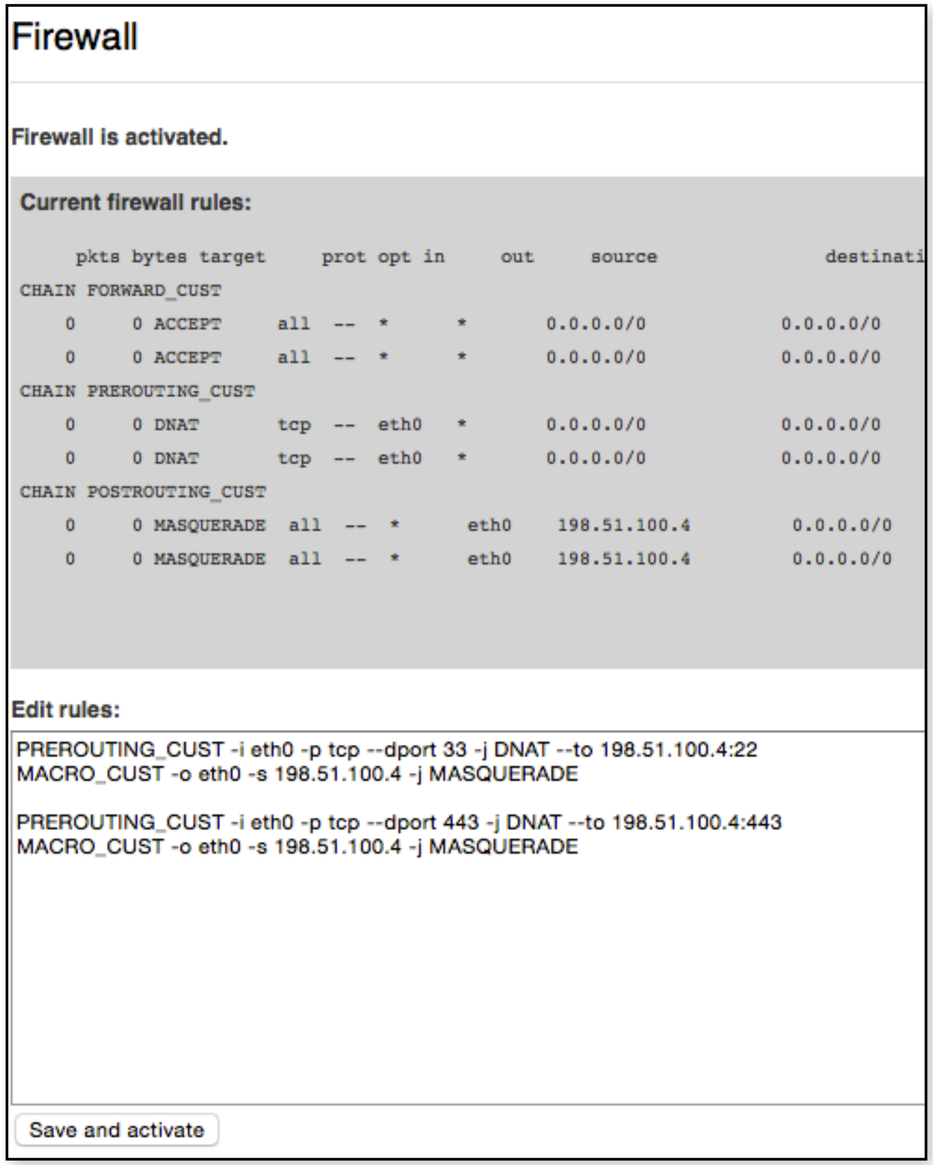
To forward arriving Web Application traffic to the WAF container uses the same technique as was shown for accessing the WAF container via Remote Shell.

Network Firewall/Security Group Rule
Allow port 443 (or 80) from your source subnet(s).

VNS3 Firewall
Enter rules to port forward incoming traffic to the Container Network and Masquerade outgoing traffic off the VNS3 Controller's outer network interface.

#Port forward port 443 to the WAF Container port 443
PREROUTING_CUST -i eth0 -p tcp -s 0.0.0.0/0 --dport 443 -j DNAT --to <WAF Container Network IP>:443

#Return the traffic back from the WAF Container via the VNS3 Controller
MACRO_CUST -o eth0 -s <WAF Container Network IP> -j MASQUERADE



For Developers / DevOps approach

Getting the WAF container source

The Docker image source is distributed as a Dockerfile along with accompanying config files.

To get the source:

```
git clone https://github.com/cohesivenet/dockerfiles.git

cd haproxy15-ssl-ssh-waf-syslog
```

SSH access

Containers launched from the image that will be built use the included authorized_keys file to specify who can gain access to the container (as root).

Insert appropriate public keys e.g.:

cp ~/.ssh/id_rsa.pub authorized_keys

cat ~/.ssh/my_other_key.pub >> authorized_keys

If you need to generate a key then:

ssh-keygen -t rsa

TLS certificates

Nginx in the container is configured to listen on port 443 and forward to the local HAproxy load balancer. The ssl.crt and ssl.key files should be replaced with suitable certificates (self signed or generated by a CA).

To generate your own private key:

```
openssl genrsa -des3 -out ssl.key 1024
```

To remove the passphrase:

```
cp ssl.key ssl.key.bak && openssl rsa -in ssl.key.bak -out ssl.key
```

To generate a Certificate Signing Request (CSR):

```
openssl req -new -key ssl.key -out ssl.csr
```

To then make a certificate:

```
openssl x509 -req -days 3650 -in ssl.csr -signkey ssl.key -out ssl.crt
```

Making a custom WAF image

A customised Docker image can be built using:

```
sudo docker build -t cohesivenet/waf-custom .
```

The tag ‘cohesivenet/waf-custom’ may be replaced with something to suit your own environment and naming conventions.

To export a container image:

```
WAF_CUSTOM=$(sudo docker run -d cohesivenet/waf-custom)

sudo docker export $WAF_CUSTOM >waf_custom.tar

gzip waf_custom.tar

sudo docker kill $WAF_CUSTOM
```

Installing the custom WAF image

(Detailed screenshots of these general plugin operations found in https://cohesive.net/dnld/Cohesive-Networks_VNS3-3.5-Docker.pdf)

First copy the waf-custom.tar.gz file to a URL capable server (Object Storage, Amazon S3, local WebDaV, Dropbox, etc) that's reachable from the VNS3:turret.

Click on the "Images" item in the Container section of the VNS3 menu.
Then select "Upload Image".

Give the image a Name: e.g. waf-custom

Paste the URL for the web server holding waf-custom.tar.gz into the Image file url: box.

Click "Upload"

Running the custom WAF image

Once the Status of the imported image is Ready then click the “Action” button and select “Allocate”.

Give the container a Name: e.g. waf-custom

The command for running the container is: /usr/bin/supervisord

Click “Allocate”

Make a note of the IP Address given to the container e.g. 198.51.100.3

Routing traffic to the WAF container

Click on the Firewall item in the Connections section of the VNS3 menu.

Add firewall rules such as:

```
MACRO_CUST -o eth0 -s 198.51.100.0/28 -j MASQUERADE
PREROUTING_CUST -i eth0 -p tcp -s 0.0.0.0/0 --dport 443 -j DNAT --to 198.51.100.3:443
PREROUTING_CUST -i eth0 -p tcp -s 0.0.0.0/0 --dport 2233 -j DNAT --to 198.51.100.3:22
```

..where 198.51.100.3 is the IP of the container once allocated. Then click “Save and activate"

SSH is now available onto the container (on port 2233 of the VNS3:turret)

Customised WAF Rules

The WAF image and containers started from it come with the OWASP ModSecurity Core Rule Set.

To add, remove or modify rules SSH into a running container and edit /etc/nginx/modsecurity.conf

One critical decision is the value of “SecRuleEngine” at the top of modsecurity.conf. In the Cohesive github files it defaults to “SecRuleEngine On” - which then has the WAF block traffic if a rule is triggered. The alternative is an alert by setting it to “SecRuleEngine DetectionOnly”.

Once a suitable rule set is created it can be cloned elsewhere by exporting the modified container from VNS3.

Resources

Questions or Corrections for this document: support@cohesive.net

Questions about configuring the WAF elements effectively: support@cohesive.net

More about the ModSecurity WAF rule set:
https://www.modsecurity.org/