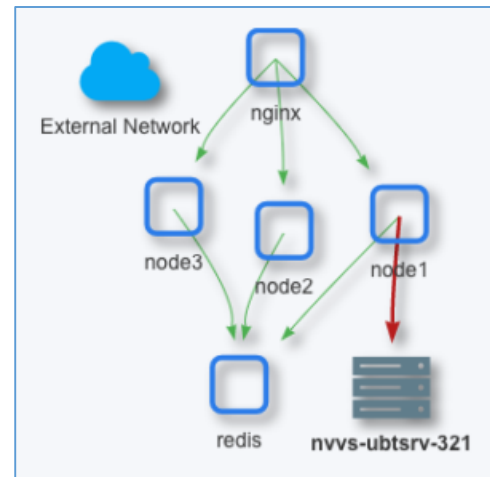


Sample Applications and Test Plan

Overview

The NeuVector Run-time Container Security Solution enables you to:

- Discover running applications, services and processes, and apply the built-in security policies for them
- Monitor running containers for violations, threats, and vulnerabilities, and
- Protect your containers from suspicious activity with no manual policies required.



This guide will help you to try out NeuVector to learn its operations and to test various functions with two sample applications. The sample applications described in this document are:

1. **Dual Host Multi-Node Web Application.** This is a Nginx load balancer and Redis database running on one host and 3 web server nodes on another host. You can use this sample application to test inter-container and inter-host violations.
2. **Single Host Scalable Web Application.** This is a Nginx load balancer, web server node and Redis database designed to scale via the Interlock service-discovery service. You can use this sample application to test NeuVector security features as well as see how it responds to dynamic scaling of webserver node containers.

Install the NeuVector Components

This guide assumes that you have already pulled the NeuVector 'allinone' and 'enforcer' images from the NeuVector Docker Hub private registry. You should also have deployed the containers and successfully logged into the NeuVector Console (default port 8443). For the Dual Host application deploy the 'allinone' container on one host and the 'enforcer' on the other. The Single Host application only requires the 'allinone' on that host.

Sample Applications and Test Plan

If you need instructions on deploying NeuVector you can pull the `neuvector/docs` image from the registry as well, and run it to view the documentation in a local browser, using

```
$ sudo docker run -d --name docs -p 80:80 neuvector/docs
```

This guide also assumed you have `docker-compose` installed.

Dual Host Multi-Node Application

You can view the instructions and required files from the '[nvbeta](#)' repository on Github. The application container images are located on the '[nvbeta](#)' public repository on Docker Hub for the [nginx image](#) and the [node image](#).

Install and run the dual host application

1. On both hosts, pull the files for the [dual node](#)

```
$ git clone https://github.com/nvbeta/nodejs
```
2. Navigate to the `/nodejs/dual` directory
3. Edit the `start_nginx_redis.sh` file to enter the IP address of your host. The Nginx 'add-host' should be the address Host 2. The edit the `start_node.sh` file, with the redis 'add_host' for the 3 nodes being the address of Host 1.
4. Start services
 - a. On host one

```
$ ./start_nginx_redis.sh
```
 - b. On host two

```
$ ./start_node.sh
```
5. Browse to port 1080 on host one to make sure everything is running.
 - a. Refresh the browser several times to generate traffic through the load balancer to each of the nodes
6. (Optional) Install a Wordpress/MySQL application on Host 1
 - a. Create a `wordpress.yml` file using the sample in the Appendix

Sample Applications and Test Plan

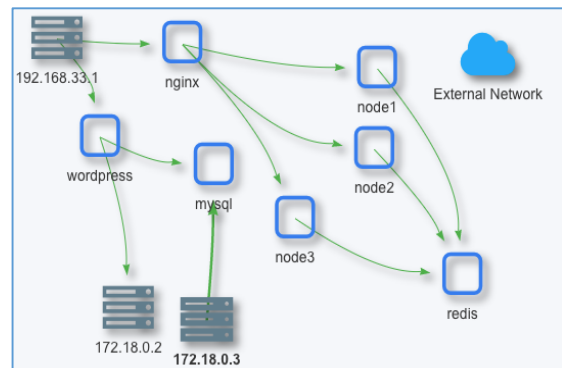
- b. Install and run Wordpress and MySQL

```
$ sudo docker-compose -f wordpress.yml up -d
```

Try Out NeuVector

Here are suggestions for testing NeuVector with your running sample application:

- **Discover new services.** In the Dashboard (default port 8443) you'll see the Containers and Services discovered. You can click on Services to see the Network Activity map and conversations between each container.
 - Refresh your browser on port 80 to increase the counter and generate traffic through the nginx load balancer. Then switch to the NeuVector Console and refresh it to see the recent connections.
 - You can also move the objects around in the Network Activity map to make it more readable
 - (Optional) If you've also installed the Wordpress app, generate traffic to it from your browser on port 2080 to see NeuVector discover the service.
- **Review security policy.** Based on application behavior (generating connections through the load balancer to the web server to the database), NeuVector automatically creates Services, Groups and Rules for allowed (normal) application behavior. Once you are happy that the normal application behavior has been discovered by NeuVector, you can switch services to Monitor mode from the Services menu.



Sample Applications and Test Plan

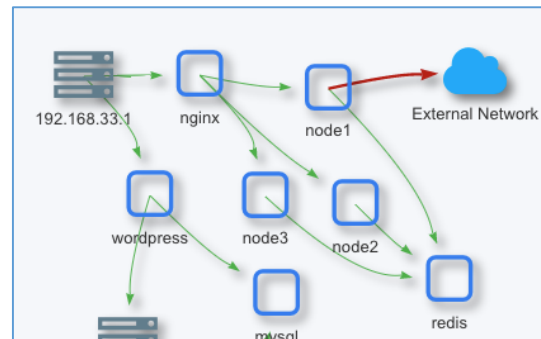
Rules						
Drag and drop rules to change the order of matching.						
New Rules Modified Rules Learned Rules						
Id	From	To	Applications	Ports	Deny/Allow	Actions
10001	nv.wordpress:latesnv.mysql:5.7	MySQL	MySQL	any	<input checked="" type="checkbox"/>	+ ×
10002	nv.wordpress:latesnv.Host:172.18.0.2	MySQL	MySQL	any	<input checked="" type="checkbox"/>	+ ×
10003	nv.nvbeta.nginx	nv.nvbeta.node	HTTP	any	<input checked="" type="checkbox"/>	+ ×
10004	nv.nvbeta.node	nv.redis:latest	Redis	any	<input checked="" type="checkbox"/>	+ ×

- Monitor for violations.** Switch NeuVector into Monitor mode to start monitoring for violations. Below are several violations to try out. After each one, you can refresh the NeuVector Console in Network Activity to verify that the violation has been detected. Violations will show in red and you can click on the red arrow to see details.

Violations to Try

- Connect externally from Node 1. Try to connect to an external website like www.google.com, using


```
$ sudo docker exec node1 curl www.google.com
```

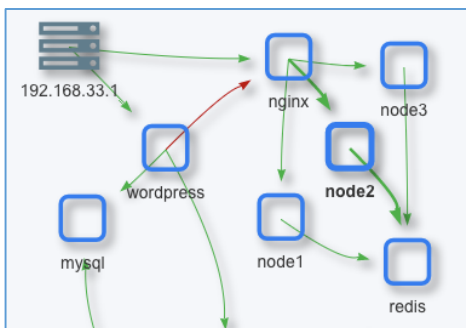


- SSH from Node1 (on Host 2) to Host 1, using

```
$ sudo docker exec node1 nc <Host_1 IP> 22
```

- Connect from Node1 (on Host 2) to nginx (on Host 1), using

```
$ sudo docker exec node1 curl <Host_1 IP>:80
```



- (Optional) On Host 1 try to connect from Wordpress to nginx, using

```
$ sudo docker exec wordpress curl <Host_1 IP>:80
```

This demonstrates how a compromised container can try to access another application.

Sample Applications and Test Plan

- Protect containers.** Now switch all services from Monitor mode to Protect mode from the Services menu. In this mode, violations will be blocked (denied). First, clear each violation by clicking on it and selecting Clear in the popup. Run the same violations as you did in Monitor mode, and refresh the Console to see them. You should see violations marked with a red arrow, like before, but with an 'x' in them indicating they were blocked.
- Quarantine a container.** From the Network Activity menu you can select any node, then click on the Quarantine button to prevent any connections to that node. A red circle around the node will show visually that it is quarantined. You can try to simulate normal connections to the node and it will be blocked.
- Run a vulnerability scan.** Go to the Vulnerabilities menu and turn on Auto-Scan. Now, all running containers will be scanned for vulnerabilities, and results shown for each container.

Name	Node	Image	Status	High	Medium	Time
redis	vagr...	redis:latest	Finished	5	8	Aug 15, 2016 ...
node3	vagr...	nvbeta/node	Finished	0	42	Aug 15, 2016 ...
node2	vagr...	nvbeta/node	Finished	0	42	Aug 15, 2016 ...
node1	vagr...	nvbeta/node	Finished	0	42	Aug 15, 2016 ...
mysql	vagr...	mysql:5.7	Finished	2	4	Aug 15, 2016 ...
wordpress	vagr...	wordpress:lat...	Finished	6	12	Aug 15, 2016 ...
nginx	vagr...	nvbeta/nginx	Finished	12	30	Aug 15, 2016 ...

Scanned: 7 Scanning: 0 Scheduled: 0 Number Of Scanners: 1 Auto Scan Scan

Single Host Scalable Application – Shows Auto-Scaling of NeuVector

You can view and download the instructions and docker-compose.yml file from the ['nvbeta'](#) repository on Github. The container images are located on the ['nvbeta'](#) public repository on Docker Hub for the [nginx image](#) and the [node image](#).

Sample Applications and Test Plan

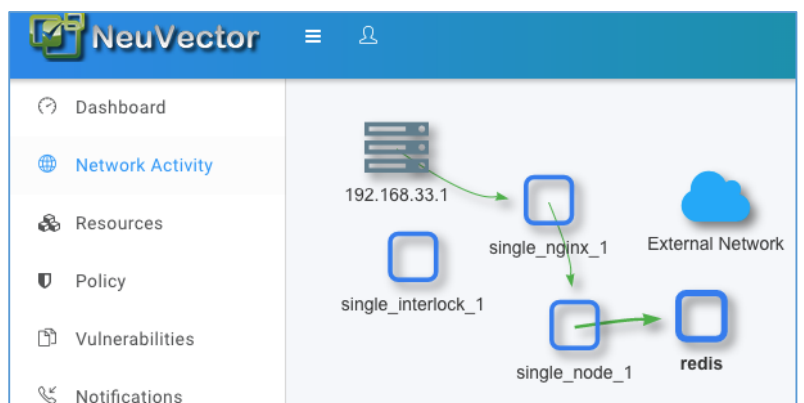
Install and run the single host application:

1. Pull the files for the [single node](#)
`$ git clone https://github.com/nvbeta/nodejs`
2. Navigate to the /nodejs/single directory
3. Edit the config.toml file: change `_BackendOverrideAddress_`'s value to the IP of your host
4. Start services
`$ sudo docker-compose up -d`
5. Browse to port 80 on host one to make sure everything is running
6. (optional) Scale up or down the number of web server nodes, using
`$ sudo docker-compose scale node=x` (where x is the number of nodes to scale up to or down)

Try Out NeuVector

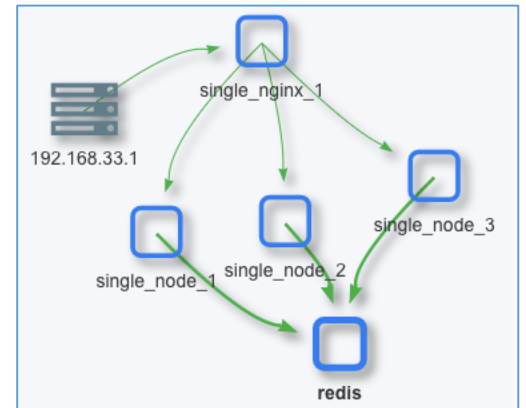
Here are suggestions for testing NeuVector with your running sample application. Log into the NeuVector Console (default port 8443), and:

- **Discover new services.** In the Dashboard you'll see the Containers and Services discovered. You can click on Services to see the Network Activity map and conversations between each container.
 - Refresh your browser on port 80 to increase the counter and generate traffic through the nginx load balancer. Then switch to the NeuVector Console and refresh it to see the recent connections.



Sample Applications and Test Plan

- **Scale up or down the number of nodes.** You can see NeuVector discover all running containers as you scale up or down, using `$ sudo docker-compose scale node=x` (where x is the number of nodes to scale up to or down – try 3 to start with)
- You can also move the objects around in the Network Activity map to make it more readable.
- **Review security policy.** Based on application behavior (generating connections through the load balancer to the web server to the database), NeuVector automatically create Services, Groups and Rules for allowed (normal) application behavior. Once you are happy that the normal application behavior has been discovered by NeuVector, you can switch all services to Monitor mode from the Services menu.
- **Monitor for violations.** Switch services into Monitor mode to start monitoring for violations. Below are several violations to try out. After each one, you can refresh the NeuVector Console in Network Activity to verify that the violation has been detected. Violations will show in red and you can click on the red arrow to see details.



Violations to Try

1. Connect externally from node. Try to connect to an external website like

www.google.com, using

```
$ sudo docker exec single_node_1 curl www.google.com
```

2. SSH from node to Host 1, using

```
$ sudo docker exec single_node_1 nc <Host_1 IP> 22
```

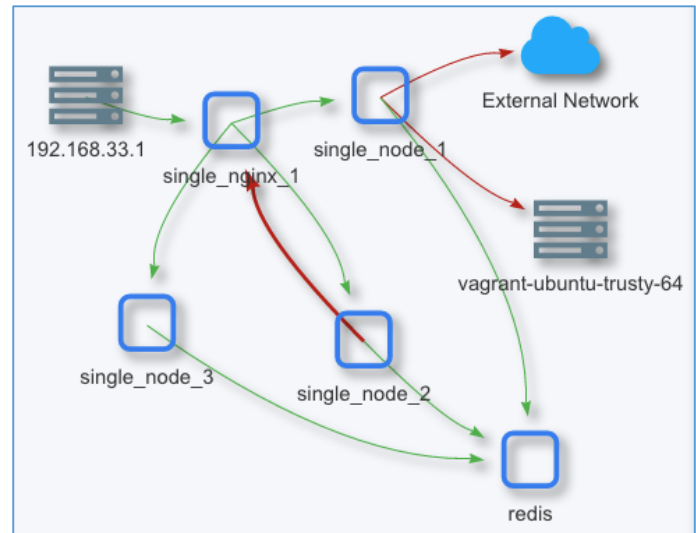
3. Connect from any node to nginx using

```
$ sudo docker exec single_node_2 curl 192.168.33.30:80
```

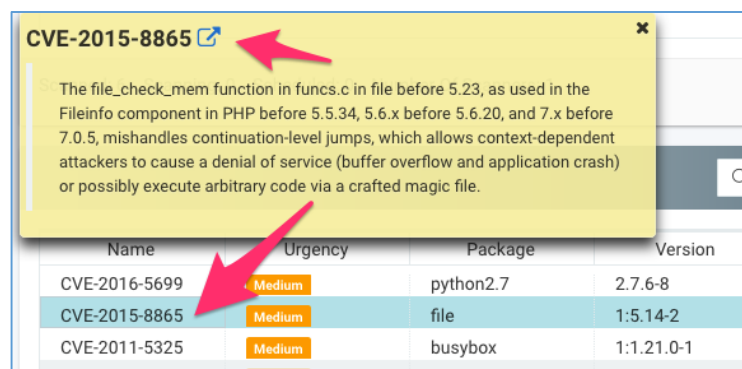
Sample Applications and Test Plan

And you can see that, although each of these connections were successful, they were detected as violations by NeuVector.

(Hint: you can click on each arrow to get more details, and click the Clear button to clear any red violations)



- Protect containers.** Now switch services from Monitor mode to Protect mode in the Services menu. In this mode, violations will be blocked (denied). First, clear each violation by clicking on it and selecting Clear in the popup. Run the same violations as you did in Monitor mode, and refresh the Console to see them. You should see violations marked with a red arrow, like before, but with an 'x' in them indicating they were blocked.
- Quarantine a container.** From the Network Activity menu you can select any node, then click on the Quarantine button to prevent any connections to that node. A red circle around the node will show visually that it is quarantined. You can try to simulate normal connections to the node and it will be blocked.
- Run a vulnerability scan.** Go to the Vulnerabilities menu and turn on Auto-Scan. Now, all running containers will be scanned for vulnerabilities, and results shown for each container. You can click on any of the vulnerabilities detected to get more detail about it.



CVE-2015-8865

The file_check_mem function in funcs.c in file before 5.23, as used in the Fileinfo component in PHP before 5.5.34, 5.6.x before 5.6.20, and 7.x before 7.0.5, mishandles continuation-level jumps, which allows context-dependent attackers to cause a denial of service (buffer overflow and application crash) or possibly execute arbitrary code via a crafted magic file.

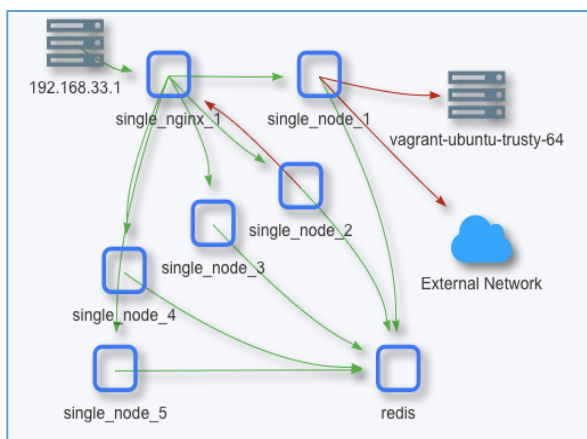
Name	Urgency	Package	Version
CVE-2016-5699	Medium	python2.7	2.7.6-8
CVE-2015-8865	Medium	file	1:5.14-2
CVE-2011-5325	Medium	busybox	1:1.21.0-1

Sample Applications and Test Plan

- **Scale Easily While In Production.** One of the coolest features of NeuVector is that the Security Policy is declarative – that is, it's describing the allowed behavior of your application, NOT just configuring firewall like rules based on container IP addresses and ports. Why is this important? You can easily scale your application without manual reconfiguring of rules.

Try scaling the number of webserver nodes from 3 to 5, using

```
$ sudo docker-compose scale node=5
```



You can see that NeuVector instantly detects the two new containers 4 and 5 and allows normal traffic through them, while still detecting and displaying the violations we generated before. Whether it's scaling up or down, or updating applications, NeuVector makes it easy and painless to manage your applications in production.

Notifications and Reporting

You can go to the Notifications menu to view Threats and Violations detected by NeuVector. You can also integrate NeuVector logs using SYSLOG for reporting. Please see the documentation for details on how to configure this.

Advanced Security Policy

Although it is not necessary to manually create Groups and Rules to protect your services, you may have situations where you want to create a custom Allow/Deny rule. You can go to the Security menu to add one or more Groups, then create a custom Rule to allow or restrict conversations between Groups. Please see the Security Policy section of the /docs for more details on creating customized policies.

Sample Applications and Test Plan

Summary

You have now discovered how NeuVector can:

- Discover running services and automatically build the security policy for protecting them
- Adapt as containers scale up or scale down, without manual configuration required
- Monitor containers for suspicious activity by detecting violations of the policy
- Protect containers by blocking connections which are not normal application behavior
- Find vulnerabilities in running containers by scanning them in real-time
- Protect containers whether they are on a single host or distributed across multiple hosts.

Questions? Feedback?

Please send any questions, suggestions or other feedback to support@neuvector.com

APPENDIX

wordpress.yml file:

```
version: '2'
services:
  db:
    image: mysql:5.7
    container_name: mysql
    volumes:
      - "./.data/db:/var/lib/mysql"
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: wordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    container_name: wordpress
    links:
      - db:mysql
    ports:
      - "2080:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_PASSWORD: wordpress
```