Workshop Steps

Prerequisites

- AWS Account
- Have a Default VPC configured
- git installed on your development machine *(for cloning the workshop repository)*
- Create an EC2 SSH Keypair
- Clone workshop repo from Github

```
git clone https://github.com/aws-samples/aws-iot-device-defender-workshop.git
```

Create your Workshop Cloudformation Stack

We will create an online development environment using AWS Cloud9. This environment will let us simulate an IoT thing, and a malicious process running on that thing. Because Cloud9 automatically sets up your AWS credentials, it will let us quickly test out different features of Device Defender and run our simulated attack.

This CloudFormation Stack will create:

- A Cloud9 IDE and associated EC2 instance (this will stand in for an IoT Thing)
- An EC2 instance to serve as a target for our simulated attack '

Steps

1. From the AWS Console, navigate to CloudFormation
2. Click the "Create Stack" button,
3. Choose "Upload a template file", and select the **workshop.yaml** from the *cloudformation* directory of the workshop GitHub repository you cloned earlier
4. Click "Next"
5. Give your stack a name: "DeviceDefenderWorkshop"
6. You can leave the AutoHibernateTimeout and InstanceType fields as they are
7. In SubnetIdentifier, choose the subnet you'd like to use
   - if you are unsure, choose the first one in the list
8. In KeyName, Select the key pair you'd like to use for ssh access to your instances
9. Click "Next" on the following screen
10. Check the "I acknowledge that AWS CloudFormation might create IAM resources." box to continue
11. Click the "Create Stack" button at the bottom of the screen
12. Wait for stack to finish, you should see "CREATE COMPLETE" in the status column after a few minutes

- Tip: *you may need to refresh your screen to see the updated status of your stack*

Login into your Cloud9 Environment

1. Go to the Cloud9 Console
2. Enter the environment "DeviceDefenderWorkshop", by clicking the "Open IDE" button

Install prerequisites

In this step, we will run a small shell script that will setup the environment so we can quickly get started learning about Device Defender

- Download Amazon CA certificates
- Install Boto3 python library for AWS
- Install AWS IoT Device SDK python package
- Install AWS IoT Device Defender Agent SDK Python Package

Steps

From a console tab towards the bottom of your Cloud9 IDE, run "bootstrap.sh" script '

```
cd workshop/scripts
```

```
./bootstrap.sh
```

Create your AWS IoT Thing

In this step, we will run a Python script that will automate creating an AWS IoT Thing, this will be the thing that we simulate in our Cloud9 instance: - An IoT Thing Group - An IoT Thing, registered in IoT Device management, placed in the group we created - An IoT Certificate, attached to your Thing - An IoT Policy attached to the Certificate - An agent_args.txt file to make running the Device Defender Agent easier

Running the Thing Provisioning script

While in the *scripts* directory, run the following:

```
./provision_thing.py
```

Setup an SNS Topic for Device Defender Violation Notifications (SNS Console)

Device Defender has the ability to send notification of a Behavior Profile violation via an SNS Topic. For this workshop, we will configure an SNS topic and enable email delivery of the notifications.

Setting up the SNS Topic

1. Navigate to the SNS Console
2. Click "Create Topic"

3. For Topic Name: “DeviceDefenderNotifications”
4. For Display Name: “DvcDefendr”
5. In the Topic Details screen for your newly created topic, click “Create Subscription”
6. For Protocol, select “Email”
7. For Endpoint enter your email address
8. Check your email, after a few moments, you should receive a confirmation email
9. Click “Confirm Subscription” link in the email

- *Note*: the sender of the email will be the same as the Display Name you entered for the topic.

Create a Target Role for Device Defender SNS Notifications

For this step, we will re-use a policy from AWS IoT Rules Engine, as it has the proper SNS policy in place.

1. Navigate to the IAM Console
2. Select Roles from the left hand menu
3. Click “Create Role”
4. In the Select type of trusted entity section, choose “AWS Service”
5. Select “IoT” as the service that will use this role
6. When you select “IoT”, a section will appear entitled “Select your use case”
7. Select “IoT”
8. Click “Next:Permissions”
9. Next you will shown a summary of attached policies, you don’t need to do anything on this screen
10. Click “Next: Tags”
11. Click “Next: Review”
12. On the Create Role screen enter “DeviceDefenderWorkshopNotification” for the Role Name
13. Click “Create Role”

Configure a behavior profile (IoT Device Defender Console)

Now that we have our simulated thing created and we have a development environment, we are ready to configure a behavior profile in device defender

1. Navigate to the Security Profiles Section of the Device Defender Console AWS IoT -> Defend -> Detect -> Security Profiles
2. Click the “Create” button
    - *Note*: If you have no Security Profiles in your account, you will see a “Create your first security profile” button instead
3. Configure parameters
4. Name: “NormalNetworkTraffic”
5. Under Behaviors:

Name: “PacketsOut”
Metric: “Packets Out”
Operator: “Less Than”
Value: “10000”
Duration: “5 minutes”

6. On the Alert Targets Page:
   SNS Topic: “DefenderWorkshopNotifications”
   Role: “DeviceDefenderWorkshopNotification”
7. Click Next
8. Attach profile to group “DefenderWorkshopGroup”
9. Click Next
10. Click Save

Start the Agent (Cloud9)

The next component of Device Defender we are going to look at is the Device Agent. The detect function of DD, can utilize both cloud-side metrics and device-side metrics. For device-side metrics, we need something that runs on the device and collects metrics and sends them to Device Defender. For this we provide reference implementations of agents that you can use as the basis for your own device-defender integration.

The reference agent we will be using today is the Python agent. It’s operation is fairly simple: periodically it wakes up, takes a sample of some basic system metrics, compiles them into a metrics report and publishes them to a reserved Device Defender MQTT Topic. From there, all processing is done automatically in the cloud by Device Defender.

Run the agent from a console tab:

```
cd scripts
```

```
python
/usr/local/lib/python2.7/sitepackages/AWSIoTDeviceDefenderAgentSDK/agent.py
@agent_args.txt
```

After a few minutes, you should see a message similar to the following:

```
Received a new message:
{"thingName":"DefenderWorkshopThing","reportId":1542697506,"status":"ACCEPTED
","timestamp":1542697506269}
```

This is an MQTT message sent from Device Defender to the agent, acknowledging acceptance of a metrics report.

Start the attacker

1. Get your Target server URL from the CloudFormation outputs from the stack you created earlier

2. In a second console tab (leave the agent running), run "ab" tool, which will generate HTTP traffic from your "device" to the target server. *Note: the trailing space is necessary here:*

```
ab -n 20000 http://YOUR_TARGET_INSTANCE_URL/
```

View Violations

AWS IoT Console

1. IoT -> Defend -> Detect -> Violations
2. View the "Now" tab to see current state
3. View the "History" tab to see how the device has changed over time

Check violation email

You should see an email from SNS indicating the violation the contents will look something like this:

```
{"violationEventTime":1542682416515,"thingName":"DefenderWorkshopThing","beha
vior":{"criteria":{"value":{"count":100},"durationSeconds":300,"comparisonOpe
rator":"less-than"},"name":"PacketsOut","metric":"aws:all-packets-
out"},"violationEventType":"alarm-
cleared","metricValue":{"count":29},"violationId":"76ef8d1dc35ed4eb802ff44568
f91097","securityProfileName":"NormalHTTPTraffic"}

--
If you wish to stop receiving notifications from this topic, please click or
visit the link below to unsubscribe:
https://sns.us-east-
1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:#########:Defend
erNotifications:######

Please do not reply directly to this email. If you have any questions or
comments regarding this email, please contact us at
https://aws.amazon.com/support
```

Confirm Violation has cleared

After approximately 10 minutes after you stop running AB, your device should no longer be in violation. *Note* You can always check your violations history tab to see how the security posture of your devices changed over time.

Cleanup

1. Delete your IoT Resources created with the provision_thing script

```
cd scripts
./provision_thing.py --cleanup
```

2. Delete your CloudFormation stack
3. Delete all other AWS resources associated with DefenderWorkshop
   - SNS Topic
   - SNS Subscription
   - IAM Role
   - Device Defender Behavior Profile

Troubleshooting

Problem: Getting an error in botocore when trying to use the aws cli

```
ImportError: cannot import name AliasedEventEmitter
```

Solution

```
sudo yum downgrade aws-cli.noarch python27-botocore
```

Problem: Cannot access URL for my target server

Solution

Make sure you append a trailing slash on the url: http://my.ec2.ip.aeces/

Extended Activities

If you have finished the main workshop materials, here are a few ideas for exploring device defender further.

Automated Response to Violations

Using SNS to Trigger a Lambda Function you can:

- Deactivate a devices's certificate
- Apply a more restrictive policy to the device's certificate
- Move the device to a special "Quarantine" thing group for further investigation

Try creating new behavior profiles acting on different metrics, and the devise ways to trigger violations.

Here are few ideas

- *Bytes In*: use "curl" or "wget" to download a large file placed on the target instance
- *TCP Connection Count*: make a large number connection to the target instance using the "ab" tool
- *Messages Sent*: Send a large number of MQTT messages to a topic