Executive summary

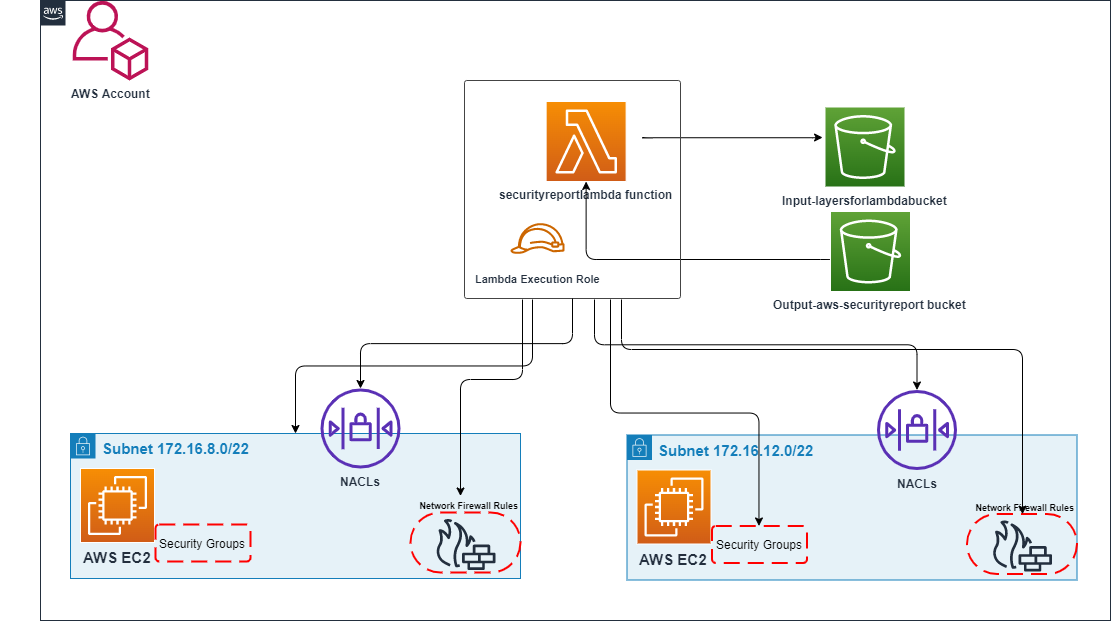
Purpose of this artifact is to enable auditing of AWS Security Groups, Network ACLS, and Network Firewall Rules in AWS accounts using AWS Lambda (serverless Service). Periodically auditing Security Groups rules, Network ACLs, and Network Firewall rules is a recommended security practice to stay in compliance with security policies and regulatory requirements. Security and Compliance is a shared responsibility between AWS and the customer. The customer assumes responsibility of managing AWS security services like Security Groups, Network ACLs, and Network Firewall. This artifact provides step by step process and the sample (baseline) code for auditing Security Groups rules, Network ACLs, and Network Firewall rules.

Challenges

Customers face following challenges while managing the rules for security groups, network acls, and network firewalls;

1. Auditing baseline security group rules, nacl (entries), and network firewall rules to AWS accounts and resources.
2. Auditing and identifying security group rules, nacl (entries), and network firewall rules pose risk.
3. Auditing and cleaning up unused or redundant security groups rules, nacl (entries), and network firewall rules.

Architecture Diagram



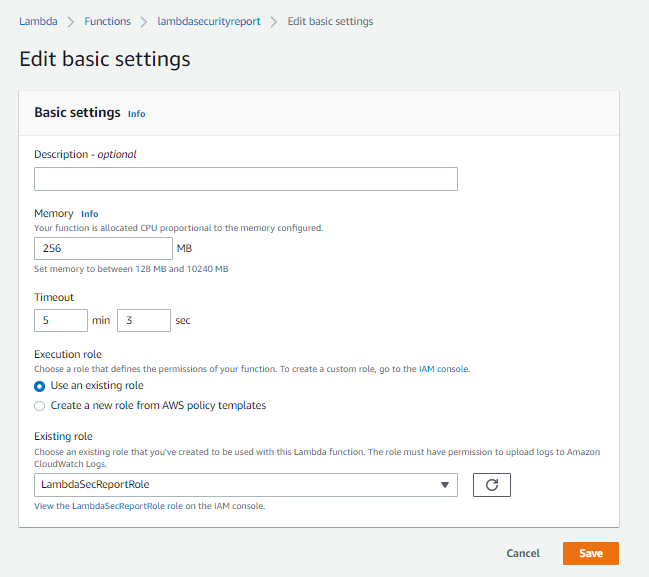
Deployment steps

Below are steps to create a Lambda Function to audit and create a report in XLSX format of the AWS accounts for exiting SGs, NACLS, and Network Firewall Rules.

1. Login to AWS Account using console.
2. From Services menu search for S3 service and choose the service.
3. Select the Region and create S3 bucket (the AWS Lambda Function.
4. Select the Region where you want to configure and create S3 bucket.
5. Download and save the sample Code (**[Github URL](https://github.com/amirdurrani/SecurityReportLambda.git))** and related modules into S3 bucket
   * Openpyxl.zip (Python library that is used to read from an Excel file or write to an Excel file)
   * Panda\_layer.zip (Python library for data analysis and visualization)
   * XLSXwrite.zip (Python module for writing files in the XLSX file format)
6. Create another S3 bucket for audit report xlsx

**Note:** the sample code user bucket named (aws-securityreport) this can be customized.

1. From Services menu search for AWS Lambda service and choose the service.
2. Create a Lambda Function and upload the Code [(GitHub URL)](https://github.com/amirdurrani/SecurityReportLambda.git)
3. Create Lambda Layers using the three modules downloaded in step#5.
4. Select deploy and configure a Test Event with a name of your preference and click on Test.

Caveats **–** Deepening on the number of entries configured for SGs, NACLs, and FW Rules there may be requirements for increasing memory and run timeout (under configuration menu) from default values.

1. The successful execution of this Lambda Function will produce output file called securityreport.xlsx (file name is customizable using sample code).

Focus on Lambda Function Code (section)

#!/bin/python

import json

import boto3

import csv

from typing import Any, Protocol

import os

import pandas as pd

import glob

import os as os

import glob as gl

import sys

from xlsxwriter.workbook import Workbook

import pathlib

================================================================================

**Section-1-# Converts output into proper JSON**

================================================================================

def convert\_to\_json(o: Any) -> Any:

t = type(o)

if t is dict:

return {k: convert\_to\_json(v) for k, v in o.items()}

elif t is list or t is set:

return [convert\_to\_json(item) for item in o]

elif t is int or t is float or t is str or t is bool or o is None:

return o

else:

return str(o)

=====================================================

**Section2-#Open Writes SG, NACL, and FW Rules in CSV Format**

==================================================================

csv\_file\_sg = open ("/tmp/sg.csv", "w") #open csv and writes SG informatioon

ccsv\_content\_sg = "Region, Group\_Name, Group\_ID, From\_Port, To\_port, CIDR, Tag\_Name\n"

csv\_file\_nacl = open ("/tmp/nacl.csv", "w")

csv\_content\_nacl = "Region, NACL\_ID, VPC\_ID, Rule\_NO, CIDR, Egress, Rule\_Action\n"

csv\_file\_fw = open ("/tmp/fw.csv", "w")

csv\_content\_fw = "Region, RuleGroupName, Protocl, Source\_IP, SourcePort, Destination\_IP, DestinationPort, Direction\n"

==========================================================================

**Section3-# for loop to get information about SGs, NACLs, and FW Rules**

==========================================================================

ec2\_client = boto3.client('ec2')

response = ec2\_client.describe\_regions()

regions\_data = response ["Regions"]

for regions in regions\_data: #main for loop to get all the regions

Name = regions["RegionName"] #Adding Region Name in new Variable called Name

#### Secuirty Groups ####

session = boto3.Session(region\_name=Name) #

client = session.client('ec2')

ec2 = boto3.resource('ec2')

response1=client.describe\_security\_groups()

sg\_data = response1 ["SecurityGroups"]

for sg in sg\_data:

# get Name tag if exists

if 'Tags' in sg:

for tag in sg['Tags']:

name\_tag\_not\_found = False

if tag['Key'] == "Name":

sg\_name\_tag = tag['Value']

else:

name\_tag\_not\_found = True

if name\_tag\_not\_found:

sg\_name\_tag = ''

else:

sg\_name\_tag = ''

if len(sg['IpPermissions']) == 0:

fromPort = ''

toPort = ''

cidr = ''

csv\_content\_sg += "{},{},{},{},{},{},{}\n".format(Name, sg["GroupName"], sg["GroupId"], fromPort, toPort, cidr, sg\_name\_tag)

else:

for portDetails in sg['IpPermissions']:

if "FromPort" in portDetails:

fromPort = portDetails.get( 'FromPort', '')

toPort = portDetails.get ('ToPort', '')

for ip in portDetails['IpRanges']:

cidr = ip['CidrIp']

csv\_content\_sg += "{},{},{},{},{},{},{}\n".format(Name, sg["GroupName"], sg["GroupId"], fromPort, toPort, cidr, sg\_name\_tag)

else:

fromPort = ''

toPort = ''

if len(portDetails['IpRanges']) == 0:

cidr = ''

csv\_content\_sg += "{},{},{},{},{},{},{}\n".format(Name, sg["GroupName"], sg["GroupId"], fromPort, toPort, cidr, sg\_name\_tag)

else:

for ip in portDetails['IpRanges']:

if len(portDetails['IpRanges']) == 0:

cidr = ''

else:

cidr = ip.get('IpRanges', '' )

csv\_content\_sg += "{},{},{},{},{},{},{}\n".format(Name, sg["GroupName"], sg["GroupId"], fromPort, toPort, cidr, sg\_name\_tag)

#### NACLs ####

session = boto3.Session(region\_name=Name) #under same mastr/main loop it

client = session.client('ec2')

ec2 = boto3.resource('ec2')

response2=client.describe\_network\_acls()

nacl\_data = response2 ["NetworkAcls"]

for nacl in nacl\_data:

vpc\_id = nacl.get("VpcId")

naclID = nacl.get("NetworkAclId")

for entry in nacl["Entries"]:

rule\_no = entry.get("RuleNumber")

cidr\_block = entry.get("CidrBlock")

egress = entry.get("Egress")

rule\_action = entry.get("RuleAction")

if not entry[ 'Egress' ]:

print(entry)

csv\_content\_nacl += "{},{},{},{},{},{},{}\n".format(Name, naclID, vpc\_id, rule\_no, cidr\_block, egress, rule\_action)

#### Firewall Rule Groups ####

session = boto3.Session(region\_name=Name)

client = session.client('network-firewall')

ec2 = boto3.resource('ec2')

response3 = client.list\_rule\_groups()

fw\_rule\_list = response3 ['RuleGroups']

for fw\_rule in fw\_rule\_list:

rule\_group = fw\_rule.get('Arn')

session = boto3.Session(region\_name=Name)

client = session.client('network-firewall')

ec2 = boto3.resource('ec2')

response4 = client.describe\_rule\_group(RuleGroupArn = str(rule\_group))

fw\_rule\_desc = response4 ['RuleGroup']

for stateRules in fw\_rule\_desc[ 'RulesSource' ][ 'StatefulRules' ]:

SourcePort = stateRules['Header']['SourcePort'].replace("[","").replace("]","")

direction = stateRules['Header']['Direction']

destPort = stateRules['Header']['DestinationPort'].replace("[","").replace("]","")

protocol = stateRules['Header']['Protocol']

source\_ip = stateRules['Header']['Source'].replace("[","").replace("]","").replace(",",";")

destiP = stateRules['Header']['Destination'].replace("[","").replace("]","")

csv\_content\_fw += "{},{},{},{},{},{},{},{}\n".format(Name, fw\_rule['Name'], protocol, source\_ip, SourcePort, destiP, destPort, direction)

====================================================

**Section4-#Write all information to add to CSV file**

====================================================

csv\_file\_sg.write (csv\_content\_sg)

csv\_file\_nacl.write (csv\_content\_nacl)

csv\_file\_fw.write (csv\_content\_fw)

#### Code ####

csv\_file\_sg.close() #Close CSV file after Writing the information obtained using the code for each service

csv\_file\_nacl.close()

csv\_file\_fw.close()

tmpPath = '/tmp/' #store the path /tmp directory under variable tmppath

os.chdir(r"/tmp/")

writer = pd.ExcelWriter('securityreport.xlsx', engine='xlsxwriter')

for csvFile in os.listdir( tmpPath ): #search for all files stored in /tmp/ that end with .csv and write each csv file as a separate tab/sheet into XLS using XSLXWirter engeine

if csvFile.endswith( 'csv' ):

fileName = pathlib.Path( csvFile )

inCsv = pd.read\_csv( os.path.join( tmpPath, csvFile ), error\_bad\_lines=False, engine ='python')

inCsv.to\_excel( writer, sheet\_name = fileName.stem, index = False )

writer.save()

===============================

**Section5-#Lambda Function**

============================

def lambda\_handler(event,context):

srcFile = '/tmp/out.xlsx'

bucketName = 'aws-securityreport'

#def uploadToS3( bucketName: str, srcFile: str ):

taskStatus = False

try:

s3\_client = boto3.client('s3')

uploadResp = s3\_client.upload\_file( srcFile, bucketName, 'securityreport.xlsx' )

print( "Upload Resp ", uploadResp )

except Exception as errMsg:

print( errMsg )

else:

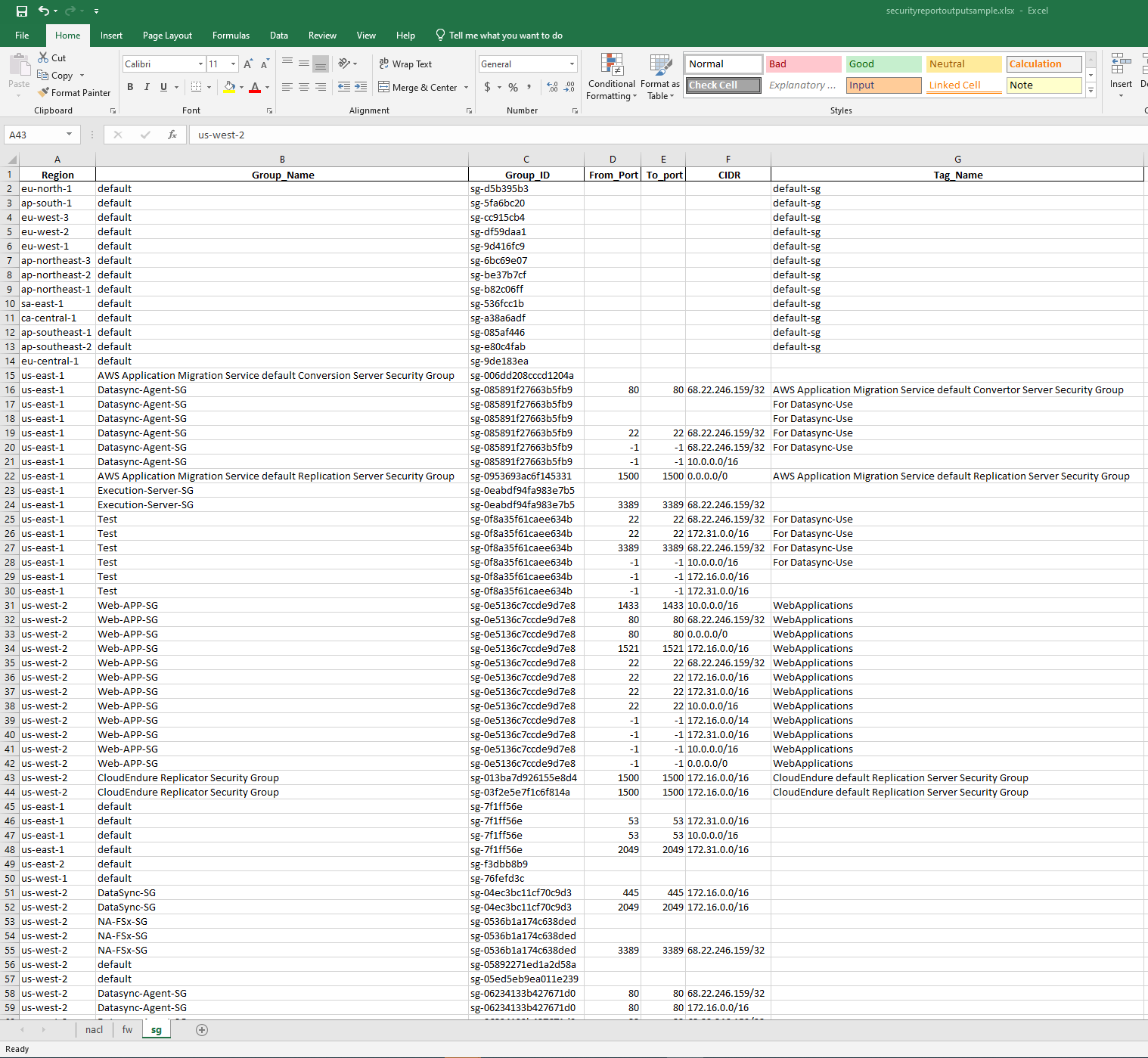
taskStatus = True

return taskStatus

#if \_\_name\_\_ == '\_\_main\_\_':

# lambda\_handler(None, None)

1. The Output file will contain 3 Tabs (sg, nacl. Fw) with audit information one each for SGs, Network ACLs, and Network Firewall Rules.



Ideas for improvement

This sample code can be customized to run from a central Networking Hub account using a Lambda Function that can be triggered when one of the following conditions is met;

1. A new account is provisioned with baseline VPC and default SGs, NACLs
2. An existing account provisions a Network Firewall
3. An existing account SGs, NACLs, or Network FW rules are deleted, modified, or added