2021

Cisco ASAv – AWS GuardDuty Integration

USER CONFIGURATION GUIDE

\

Table of Contents

Cisco ASAv – AWS GuardDuty Integration

This document explains how to deploy Cisco ASAv – AWS GuardDuty Integration solution in AWS.

Note:
Read the entire document before starting deployment. Make sure the prerequisites are met before starting deployment.

Cisco ASAv – AWS GuardDuty Integration Solution

The following sections describes the details and various components of the solution.

AWS(Amazon) GuardDuty Service

Amazon GuardDuty is a monitoring service which processes data from various sources in AWS environment (VPC Logs, CloudTrail management event logs, CloudTrail S3 data event logs, DNS logs etc.) and identifies potentially unauthorized and malicious activity in the AWS environment.

As per AWS documentation:

> "*Amazon GuardDuty is a continuous security monitoring and threat detection service that incorporates threat intelligence, anomaly detection, and machine learning to help protect your AWS resources, including your AWS accounts.*
>
> *Amazon GuardDuty is a continuous security monitoring service that analyses and processes the following Data sources: VPC Flow Logs, AWS CloudTrail management event logs, CloudTrail S3 data event logs, and DNS logs. It uses threat intelligence feeds, such as lists of malicious IP addresses and domains, and machine learning to identify unexpected and potentially unauthorized and malicious activity within your AWS environment.*
> *This can include issues like escalations of privileges, uses of exposed credentials, or communication with malicious IP addresses, or domains.*
>
> *For example, GuardDuty can detect compromised EC2 instances serving malware or mining bitcoin. It also monitors AWS account access behaviour for signs of compromise, such as unauthorized infrastructure deployments, like instances deployed in a Region that has never been used, or unusual API calls, like a password policy change to reduce password strength.*
> *GuardDuty informs you of the status of your AWS environment by producing security findings that you can view in the GuardDuty console or through Amazon CloudWatch events."*

AWS GuardDuty generates a finding whenever it detects unexpected and potentially malicious activity in your AWS environment. Based on the resource type, GuardDuty findings are categorised to EC2, IAM, S3 finding types. Also findings will have severity level associated with it. The value of the severity can fall anywhere within the 0.1 to 8.9 range, with higher values indicating greater security risk.

Severity level	Value range	Description
High	8.9 to 7.0	A High severity level indicates that the resource in question (an EC2 instance or a set of IAM user credentials) is compromised and is actively being used for unauthorized purposes.
Medium	6.9 to 4.0	A Medium severity level indicates suspicious activity that deviates from normally observed behaviour and, depending on your use case, may be indicative of a resource compromise.
Low	3.9 to 1.0	A low severity level indicates attempted suspicious activity that did not compromise your network, for example, a port scan or a failed intrusion attempt.

Note: Values 0 and 9.0 to 10.0 are currently reserved for future use.

Please refer to the AWS documentation for further details on GuardDuty Service.

Integration of GuardDuty Service with Cisco ASAv

Cisco provides set of Scripts(lambda function) and CloudFormation Template to allow the integration the AWS GuardDuty Service with Cisco ASAv (Adaptive Security Appliance Virtual).

This solution make use of the threat analysis data/results from Amazon GuardDuty (malicious IPs generating threats, attacks etc.) and feeds that information(malicious IP) to the Cisco Adaptive Security Appliance (ASAv) to protect the underlying network and applications against future threats originating from these sources(malicious IP).

This solution makes use of several AWS services, including GuardDuty, CloudWatch, Lambda, S3 Bucket, SNS, etc.
This is a complete serverless implementation (i.e. no helper VMs involved in this feature).

All this is achieved using a lambda function which is triggered by a CloudWatch event when the finding is raised by AWS GuardDuty.

- The findings detected by GuardDuty will be processed (analysed and filtered) based on pre-defined criteria like severity, connection direction attribute(INBOUND only), availability of malicious IP, etc. in the GuardDuty finding.
- Once the finding is processed, we will end up with the remote IP which caused the GuardDuty alert, this IP will be considered malicious IP.
- In next step, the network object group(s) on the ASAv(s) is created/updated with the malicious IP found after processing.

Apart from updating the network object group(s) on ASAv(s) the lambda function also:

- Creates a report file in the S3 bucket containing all the malicious IPs processed by it

- Sends a mail notification to the administrator regarding the various updates(and/or any errors) carried out by the lambda function

Please note that:

- The scope of lambda function is limited only to updating the network object group with the malicious host IP and it will be the user's responsibility to make use of this object group in ACL/ACE for blocking the traffic.
- The lambda function processes **ONLY** the findings which satisfy all conditions listed below:
    - Severity level of the GuardDuty finding is greater than equal to the configured minimum severity level.
    - The connection direction attribute is INBOUND in the GuardDuty finding. (incoming threats only)
    - The remote IP field (threat originator) is available in the GuardDuty finding.

Supported Software Platforms

This solution is applicable to the ASAv.
Although the solution is being introduced as part of 9.18.1 release, the solution software is version agnostic.
The solution uses various AWS resources and hence can be deployed only in AWS environment. But the malicious IP updates(network object groups) can be done on any ASAv (running on any platform), which is reachable by lambda via a Public IP.

Limitations

Here are some restrictions/limitations associated with the feature:

- The AWS Services used in this solution are region specific(GuardDuty, lambda, etc.), therefore solution is specific to a region. In order to use the GuardDuty findings data from various regions, region specific instances of the solution has to be deployed.
- ASAv updates are configured through CLI over SSH(No support of ASDM, CDO etc.)
- Only password based login is supported, no other authentication mechanism is supported. (SSH key based login, etc. will be considered in future)
- Regarding the encrypted passwords in the input file(ASAv configuration):
    - Encryption using the Symmetric KMS keys is only supported.
    - All the passwords must be encrypted with a single KMS key accessible to the lambda function.

How the Cisco ASAv – AWS GuardDuty Integration Solution Works

Figure 1 depicts the various components involved in the solution and how they interact with each other to complete the solution.

The AWS GuardDuty service reports a finding, based on the finding reported a CloudWatch event rule(which monitors the GuardDuty findings) triggers the AWS Lambda function.
Once the AWS Lambda functions runs, it performs the below actions:

1. Process the reported finding to verify that all the required criteria are met(severity, INBOUND connection direction, presence of malicious IP, not a duplicate finding etc.)
2. Update the network object group(s) on the ASAv(s) (as per the input configuration) with the malicious IP
3. Update the malicious IP in the report file on S3 bucket
4. Send a mail notification to the administrator regarding the various ASAv updates(and/or any errors)

The lambda function only updates the network object group(s) on the ASAv(s) and user is responsible for creating any ACL rule/entry which uses the updated network object group. Note that, this will be needed only once. Because once the ACL rule/entry is configured with the object group, future updates will update the object group and there won't be any need to update the ACL rule/entry.

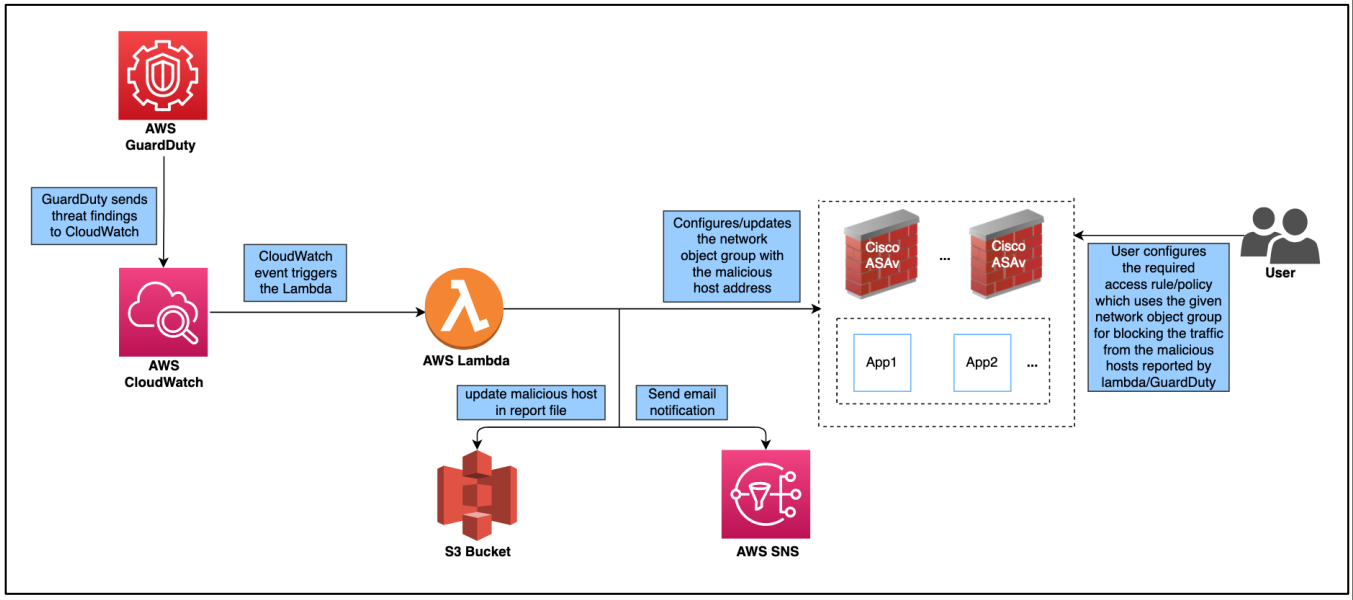

Figure 1: Cisco ASAv – AWS GuardDuty Integration Solution

Cisco ASAv – AWS GuardDuty Integration Solution Components

The following components make up the Cisco ASAv – AWS GuardDuty Integration solution.

AWS GuardDuty

AWS GuardDuty service is responsible for generating threat findings for the various AWS resources(EC2, S3, IAM, etc.) in the specific region.

AWS Simple Storage Service (S3)

S3 storage is used for storing various artifacts associated with the solution:

- Lambda function zip file
- Lambda layer zip file

- ASAv configuration input file(*.ini)
- Output report file(*.txt) containing list of malicious IPs reported

AWS CloudWatch

CloudWatch service serves two different purposes:

- **Event Rule**: Monitors the GuardDuty service for any findings reported and triggers the lambda function to process the finding.
- **Log Group**: All the logs from the lambda function are captured in the CloudWatch log group.

AWS Simple Notification Service (SNS)

Simple Notification Service (SNS) from AWS is used to publish email notifications.
The email notification contains:

- details of the GuardDuty finding successfully processed by the lambda function
- details pertaining to the updates carried on the various ASAv(s) devices by lambda function
- any major errors encountered by the lambda function

AWS Lambda Function

The lambda function is developed in Python and serves as the core of this solution.
Lambda function is triggered by the CloudWatch event rule based on GuardDuty finding.
The lambda function is responsible for:

1. Processing the reported GuardDuty finding to verify that all the required criteria are met(severity, INBOUND connection direction, presence of malicious IP, not a duplicate finding etc.)
2. Updating the network object group(s) on the ASAv(s) (as per the input configuration) with the malicious IP
3. Updating the malicious IP in the report file on S3 bucket
4. Sending a mail notification to the administrator regarding the various ASAv updates(and/or any errors)

CloudFormation Template

The CloudFormation template is used to deploy various resources required by the solution in AWS.
Following resources are created by the CloudFormation template:

- **AWS::SNS::Topic** : SNS Topic for publishing the email notifications.
- **AWS::Lambda::Function, AWS::Lambda::LayerVersion** : Lambda function and lambda layer resources using the corresponding the zip file provided.
- **AWS::Events::Rule** : CloudWatch event rule to trigger the lambda function based on the GuardDuty findings event.
- **AWS::Lambda::Permission** : Permission for CloudWatch event rule to trigger the lambda function.

- **AWS::IAM::Role, AWS::IAM::Policy** : IAM role and policy resources to allow various access permissions to the lambda function for various AWS resources involved.(S3, SNS, CloudWatch logs, KMS, etc.)

The template takes user input(parameters) to customize the deployment.

Note:
The template has limitations in validating user input, hence it is the user's responsibility to validate input during deployment.

Solution Deployment Steps

Note:
AWS Services used in this solution are region specific(GuardDuty, lambda, Dynamo DB, etc.), therefore ensure that all the AWS resources are in same region.

1. Ensure that GuardDuty Service is enabled on the AWS.
   To enable the service from AWS console:

   ```
   AWS Console --> Services --> GuardDuty --> Try GuardDuty for free /
   Get Started --> Enable GuardDuty
   ```

   Please note the below regarding the CloudWatch Events notification frequency for GuardDuty findings:

   - **Notifications for newly generated findings with a unique finding ID**
     GuardDuty sends a notification based on its CloudWatch event within 5 minutes of the finding. This event (and this notification) also includes all subsequent occurrences of this finding that take place in the first 5 minutes since this finding with a unique ID is generated.

   - **Notifications for subsequent finding occurrences**
     Frequency of notifications sent for the subsequent finding occurrences is a configurable value: 15 minutes, 1 hour, or the default 6 hours.

   Ref: CloudWatch Events notification frequency for GuardDuty

   Therefore, you might experience delay between the time when the finding is reported and the time when lambda function is invoked.

2. Download the scripts and template for the solution are from the GitHub repository:
   URL:

   Cloning the repository:

   ```
   git clone ssh://git@bitbucket-eng-bgl1.cisco.com:7999/vcb/aws-
   guardduty.git
   ```

   Repository contents:

   ```
   /
   ```

```
- README.MD
- configuration/ => ASAv Configuration file template
- images/
- lambda/        => Lambda function python files
- templates/     => CloudFormation template for deployment
```

NOTE:
Note that Cisco-provided deployment scripts and templates for this solution are provided as open source examples, and are not covered within the regular Cisco TAC support scope. Check GitHub regularly for updates and ReadMe instructions.

3. ASAv configurations

   a. Create the network object groups for the hosts reported by the GuardDuty/lambda.
   You may have to create the object group with a dummy IP in order to use this object group in a ACL rule/entry initially.
   (In the below example *12.12.12.12* is the dummy IP used)

   This needs to be created on all the ASAv(s) which will be configured to be updated by the solution.
   Also, you can create multiple network object groups on single ASAv to be updated.

   ```
   object-group network aws-gd-suspicious-hosts
      description Malicious Hosts reported by AWS GuardDuty
      network-object host 12.12.12.12
   ```

   In case you don't create this beforehand, the lambda function will create and update the malicious IPs in a network object group with the default name: **aws-gd-suspicious-hosts**

   b. Create appropriate ACL rule/entry to block the traffic using the network object group (above)
   The below example is just a sample for reference, the ACL rules may differ based on the requirement.

   ```
   access-list out-iface-access line 1 extended deny ip object-
   group aws-gd-suspicious-hosts any
   ```

   c. Create a separate user(e.g. aws-gd) on ASAv to be used by lambda function for configuration updates.
   Below is just a sample, you may create the username and password as per your choice.

   ```
   username aws-gd password MyPassword@2021 privilege 15
   username aws-gd attributes
   service-type admin
   ```

   Note:
   Please note that, latest ASAv versions will force the new user to reset the password on first login to the ASAv. Therefore you will have to update the CLI configured password.

So after configuring the new user and password on ASAv CLI, confirm the login to ASAv with new user and update the password (when prompted). Also, verify that enable password works as expected.

Finally configure this updated user password and enable password in ASAv configuration input file.

d. Encrypt the password(s) and enable password(s) (*optional*)
This is required only if you are providing encrypted password in the input configuration. You may also provide password in plain-text.
Please note that in case encrypted passwords are configured:

i. All the passwords(and enable passwords) must be encrypted with a single KMS key accessible to the lambda function.(in same region)
You may use the below command to generate encrypted password:

```
aws kms encrypt --key-id <KMS-ARN> --plaintext <password>
```

Encrypted Password is value of **`CiphertextBlob`** in above command output.
You will need the AWS CLI installed and configured to run the above command.
Please refer the AWS CLI documentation for more details:
AWS Command Line Interface

ii. Both Password and Enable Password must be encrypted

iii. Encryption using the Symmetric KMS keys is only supported.

iv. The ARN of the KMS key should be provided as the parameter during deployment of CloudFormation template

e. Create the ASAv configuration input file.
Refer the template config file template available under configuration folder:
`configuration/asav-config-input.ini`

Below are the various ASAv details needed:

```
[asav-1]  ==> Section name: Unique ASAv Identifier (within
the file)
public-ip=<Public IP of ASAv>
username=<Username for login on ASAv>
password=<Password (plaintext/encrypted using KMS)>
enable-password=<Enable Password (plaintext/encrypted using
KMS)>
object-group-name=<Network Object Group Name(s) to be
updated with malicious host IP (comma separated values)>
```

Note:

i. Before configuring the credentials (username, password, enable password) for various ASAv devices in the configuration file, please ensure that login works properly using these credentials.
ii. Ensure that the entry/section for an ASAv device is added only once in the configuration file, multiple entries for same device may cause race conditions and nondeterministic behaviour as lambda will work on to update all the devices simultaneously.
In case multiple network object group updates are needed on single device, same can be configured as comma separated values for the **object-group-name** attribute.

4. Prepare the lambda function zip file.
Create the zip file using the python files in lambda folder: `lambda/*.py`

```
$ cd lambda
$ zip asav-gd-lambda.zip *.py
adding: asav.py (deflated 73%)
adding: aws.py (deflated 74%)
adding: main.py (deflated 70%)
adding: utils.py (deflated 61%)
$
```

5. Prepare the lambda layer zip file.
The lambda layer file: `asav-gd-lambda-layer.zip` can be created on a Linux environment, such as Ubuntu 22.04 with Python 3.9 installed.

```
$ mkdir -p layer
$ virtualenv -p /usr/bin/python3.9 ./layer/

$ source ./layer/bin/activate

$ pip3.9 install cffi==1.15.0
$ pip3.9 install cryptography==37.0.2
$ pip3.9 install paramiko==2.7.1

$ mkdir -p ./python/.libs_cffi_backend/
$ cp -r ./layer/lib/python3.9/site-packages/* ./python/

$ zip -r asav-gd-lambda-layer.zip ./python
```

Note: Install python3.9 and its dependencies(as shown below) for creating the layer.

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository ppa:deadsnakes/ppa
$ sudo apt install python3.9
$ sudo apt install python3-virtualenv
$ sudo apt install zip
$ sudo apt-get install python3.9-distutils
$ sudo apt-get install python3.9-dev
$ sudo apt-get install libffi-dev
```

6. Create a S3 bucket and upload all the artifacts.
Upload the below artifacts created in previous steps.
   a. ASAv Configuration file: ***asav-config-input.ini***
   b. Lambda layer zip file: ***asav-gd-lambda-layer.zip***
   c. Lambda function zip file: ***asav-gd-lambda.zip***

7. Deploy the CloudFormation template

   a. Input Parameters for CloudFormation template
   The following input parameters should be collected prior to deployment.

Sr. No.	Parameter	Description
1.	Deployment name*	This will be used as prefix for all the resources created by the cloud formation template. e.g. *cisco-asav-gd*
2.	Minimum severity level of GD finding*	Minimum GuardDuty findings severity level to be considered for processing[1.0 to 8.9]. Any GuardDuty finding reported with lesser severity than this will be ignored. Severity classification, Low: 1.0 to 3.9, Medium: 4.0 to 6.9, High: 7.0 to 8.9. e.g. *4.0*
3.	Administrator email ID*	Administrator email address to receive notifications regarding ASAv updates done by lambda function. e.g. *abc@xyz.com*
4.	S3 Bucket name*	S3 Bucket name containing the files (lambda function zip, lambda function zip and ASAv details file). e.g. *asav-gd-bucket*
5.	S3 Bucket folder/path prefix	S3 Bucket path/folder containing the config files, leave empty if there is no folder. e.g. *"" or "cisco/asav-gd/"*
6.	Lambda layer zip file name*	Lambda layer zip file name. e.g. *asav-gd-lambda-layer.zip*
7.	Lambda function zip file name*	Lambda function zip file name. e.g. *asav-gd-lambda.zip*
8.	ASAv configuration file name*	'*.ini' file containing ASAv(s) configuration details. (Public IP, username, password, enable password, network object group names, etc.) e.g. *asav-config-input.ini*
9.	ARN of KMS key used for password encryption	ARN of an existing KMS (AWS KMS key used for password encryption). Leave empty in case plain text passwords are provided in the ASAv configuration input file. If specified, all the passwords and enable passwords mentioned in the ASAv configuration input file must be encrypted. The Passwords must be encrypted only using the specified ARN. Generating encrypted passwords: aws kms encrypt --key-id <KMS ARN> --plaintext <password> e.g. *arn:aws:kms:<region>:<aws-account-id>:key/<key-id>*
10.	Enable/Disable debug logs*	Enable/Disable lambda function debug logs in CloudWatch. e.g. *enable or disable*

'*': denotes the mandatory parameters.

b. Deploy the stack
After all of the prerequisites are completed for deployment, you can create the AWS CloudFormation stack.
Use the template file in the target directory:
templates/cisco-asav-gd-integration.yaml
Provide the parameters as collected in previous step.

Deployment:

```
AWS Console --> Services --> CloudFormation --> Stacks -->
Create stack (with new resources) --> Prepare template(Template
is ready) --> Specify template --> Template source(Upload a
template file) --> <update parameters> --> Create Stack
```

Please refer AWS Documentation for deployment of CloudFormation template. AWS Documentation

c. Subscribe to the email notifications
Once the CloudFormation template is successfully deployed, you will receive a SNS notification mail(on configured email ID) requesting you to subscribe. You should subscribe to the same using the link provided in the mail.

Deployment Verification (*optional*)

Once the CloudFormation Deployment is completed, you may verify the deployed solution using below steps.

1. Fetch the GuardDuty detector ID from AWS Console

```
AWS Console --> Services --> GuardDuty --> Settings --> About
GuardDuty --> Detector ID
```

2. In case of newly enabled service the findings list in GuardDuty console will be empty, Therefore, generate the sample GuardDuty finding using the below cmd(s).

```
aws guardduty create-sample-findings --detector-id <detector-id> --
finding-types <GuardDuty-Finding types to be generated>

aws guardduty create-sample-findings --detector-id <detector-id> --
finding-types UnauthorizedAccess:EC2/MaliciousIPCaller.Custom
```

Note that the severity of the sample finding should be greater than the minimum severity level. *UnauthorizedAccess:EC2/MaliciousIPCaller.Custom* finding is raised with severity level 5 (Medium).

You will need the AWS CLI installed and configured to run the above command. Please refer the AWS CLI documentation for more details: AWS Command Line Interface

3. You should see the sample finding in the findings list on GuardDuty console. The Sample findings will contain the prefix '[sample]'. You may check the sample finding details. (attributes: like connection direction, remote IP etc.)

4. Now wait for the lambda function to run, it may take up to 5 mins for GuardDuty finding CloudWatch to trigger the lambda.

5. Once the lambda function completes its run, you may verify the below:

    1. You should receive the mail with the details regarding GuardDuty finding received and ASAv updates done by the lambda.

    2. You may verify that the network object group(s) are updated on the configured ASAv(s) with the malicious IP reported from the sample GuardDuty finding.

    3. You should verify the report file in the generated in the S3 bucket. It should contain the malicious IP entry from the sample GuardDuty finding.
    Report file name: *<deployment-name>-report.txt*

    4. You may also verify the lambda logs in CloudWatch console.
    CloudWatch Log Group Name: *<deployment-name>-lambda*

    ```
    AWS Console --> Services --> CloudWatch --> Logs --> Log groups
    --> select the log group
    ```

6. Sample finding data clean-up
    Once the above details are verified, you must clean up the data generated by the sample finding
    1. On GuardDuty console Archive the sample finding
    ```
    AWS Console --> Services --> GuardDuty --> Findings --> Select
    the finding --> Actions --> Archive
    ```

    2. Clean up the ASAv(s) by removing the malicious IP added in the network object groups

    3. Clean up the report file in S3 bucket. You may update the file by removing the malicious IP reported by the sample finding.

Updating Deployment Configuration

Post deployment you may update the various deployment configurations.
Please refer the table below for more details on the same.

Sr. No.	Parameter	Description
1.	ASAv configuration file name	Update the file in S3 bucket. You may update the file with same name as previous one.

Sr. No.	Parameter	Description
		In case filename is modified, then update the Parameter via **'Update stack'** action on AWS Console.
2.	Lambda layer zip file name	Update the zip file in S3 bucket with a new name and then update the Parameter via **'Update stack'** action on AWS Console.
3.	Lambda function zip file name	Update the zip file in S3 bucket with a new name and then update the Parameter via **'Update stack'** action on AWS Console.
4.	Minimum severity level of GD finding	Use the **'Update stack'** action on AWS Console to update the parameter value.
5.	Administrator email ID	Use the **'Update stack'** action on AWS Console to update the parameter value. You may also add/update subscriptions via SNS service console.
6.	ARN of KMS key used for password encryption	Use the **'Update stack'** action on AWS Console to update the parameter value.
7.	Enable/Disable debug logs	Use the **'Update stack'** action on AWS Console to update the parameter value.

Update stack action for an already deployed stack is available on AWS Console.

```
AWS Console --> Services --> CloudFormation --> Stacks --> <Stack name> -->
Update (Update Stack) --> Prepare template --> Use current template -->
Next --> <update parameters> --> Update Stack
```

Note:
It is not recommended to update the *S3 Bucket* and/or *S3 Bucket folder/path prefix* values post deployments.
But still if you need to change it, then:

a. Create the new bucket and folder(if needed)

b. Ensure that all the artifacts are copied from old bucket to the new bucket (within new folder):
   i. ASAv Configuration file: *asav-config-input.ini*
   ii. Lambda layer zip file: *asav-gd-lambda-layer.zip*
   iii. Lambda function zip file: *asav-gd-lambda.zip*
   iv. Output report file: *<deployment-name>-report.txt*

c. Use the 'Update stack' action on AWS Console to update the parameter value.
   i. S3 Bucket name
   ii. S3 Bucket folder/path prefix

Troubleshooting and Debugging

AWS CloudFormation Console

AWS CloudFormation Console allows you to create, monitor, update and delete stacks directly from your web browser.
You may verify the below details from the CloudFormation Console by navigating to the stack and selecting the stack:

- The status of Stack creation/update. You may check the events tab for details in case of any failures.
- The values input parameters to your CloudFormation stack.
- Status of various AWS resources created by the stack.

AWS Lambda Console

AWS Lambda console allows you to access various lambda function related details, resources and actions.
On this console you may verify the inputs to Lambda Functions, on the Lambda Functions environment variables tab under configuration section.
Lambda Function Name: ***<deployment-name>-lambda***

Amazon CloudWatch Logs

You can view logs of Lambda function on the AWS CloudWatch Console.
The lambda function logs are available under log group.
CloudWatch Log Group Name: ***<deployment-name>-lambda***

AWS S3 Console

You can verify the S3 Bucket, folder and its contents via the AWS S3 Console.
You can check for any anomaly in the input provided in the CloudFormation template against the actual objects name in the S3 bucket.

To learn more about the service console capabilities, please refer to the AWS service specific documentations.

References

- AWS GuardDuty
- CloudWatch Events notification frequency for GuardDuty
- AWS CloudFormation
- AWS Command Line Interface
- Cisco ASA CLI Configuration Guide
- Cisco ASAv Resources