

Building well-architected serverless applications: Implementing application workload security – part 2

by Julian Wood | on 13 JUL 2021 | in [Amazon EventBridge](#), [Amazon RDS](#), [AWS CloudFormation](#), [AWS CloudTrail](#), [AWS Identity and Access Management \(IAM\)](#), [AWS Lambda](#), [AWS Secrets Manager](#), [Serverless](#) | [Permalink](#) | [Share](#)

This series of blog posts uses the [AWS Well-Architected Tool](#) with the [Serverless Lens](#) to help customers build and operate applications using best practices. In each post, I address the serverless-specific questions identified by the Serverless Lens along with the recommended best practices. See the [introduction post](#) for a table of contents and explanation of the example application.

Security question SEC3: How do you implement application security in your workload?

This post continues [part 1](#) of this security question. Previously, I cover reviewing security awareness documentation such as the [Common Vulnerabilities and Exposures](#) (CVE) database. I show how to use [GitHub](#) security features to inspect and manage code dependencies. I then show how to validate inbound events using [Amazon API Gateway](#) request validation.

Required practice: Store secrets that are used in your code securely

Store secrets such as database passwords or API keys in a secrets manager. Using a secrets manager allows for auditing access, easier rotation, and prevents exposing secrets in application source code. There are a number of AWS and third-party solutions to store and manage secrets.

AWS Partner Network (APN) member [Hashicorp](#) provides [Vault](#) to keep secrets and application data secure. Vault has a centralized workflow for tightly controlling access to secrets across applications, systems, and infrastructure. You can store secrets in Vault and access them from an [AWS Lambda](#) function to, for example, access a database. You can use the [Vault Agent for AWS](#) to authenticate with Vault, receive the database credentials, and then perform the necessary queries. You can also use the [Vault AWS Lambda extension](#) to manage the connectivity to Vault.

[AWS Systems Manager Parameter Store](#) allows you to store configuration data securely, including secrets, as parameter values.

[AWS Secrets Manager](#) enables you to replace hardcoded credentials in your code with an API call to Secrets Manager to retrieve the secret programmatically. You can protect, rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. You can also generate secure secrets. By default, Secrets Manager does not write or cache the secret to persistent storage.

Parameter Store integrates with Secrets Manager. For more information, see [“Referencing AWS Secrets Manager secrets from Parameter Store parameters.”](#)

To show how Secrets Manager works, deploy the solution detailed in “[How to securely provide database credentials to Lambda functions by using AWS Secrets Manager](#)”.

The [AWS CloudFormation](#) stack deploys an [Amazon RDS](#) MySQL database with a randomly generated password. This is stored in Secrets Manager using a [secret resource](#). A Lambda function behind an API Gateway endpoint returns the record count in a table from the database, using the required credentials. Lambda function environment variables store the database connection details and which secret to return for the database password. The password is not stored as an environment variable, nor in the Lambda function application code.

Environment variables (5) Edit	
The environment variables below are encrypted at rest with the default Lambda service key.	
Key	Value
HELPER_FUNCTION_ARN	arn:aws:lambda:us-east-1:[REDACTED]:function:demo-ssm-LambdaRDSManageConnections-GKFwDMekk7UG
RDS_DB_NAME	demodb2
RDS_HOST	rotation-instance.cbrfoqimfhmm.us-east-1.rds.amazonaws.com
RDS_USERNAME	admin
SECRET_NAME	arn:aws:secretsmanager:us-east-1:[REDACTED]:secret:MyRDSInstanceRotationSecret-PkhGUveWw39z-kZhI7B

Lambda environment variables for Secrets Manager

The application flow is as follows:

1. Clients call the API Gateway endpoint
2. API Gateway invokes the Lambda function
3. The Lambda function retrieves the database secrets using the Secrets Manager API
4. The Lambda function connects to the RDS database using the credentials from Secrets Manager and returns the query results

View the password secret value in the Secrets Manager console, which is randomly generated as part of the stack deployment.

arn:aws:secretsmanager:us-east-1: [REDACTED]:secret:
MyRDSInstanceRotationSecret-nhuslUsencU9-u8Dkrx [REDACTED]

Secret details

Encryption key
DefaultEncryptionKey

Secret description

This is my rds instance secret

Secret name
MyRDSInstanceRotationSecret-nhuslUsencU9

Secret ARN
arn:aws:secretsmanager:us-east-1:[REDACTED]:secret:MyRDSInstanceRotationSecret-nhuslUsencU9-u8Dkrx

Secret value [Info](#)

Retrieve and view the secret value.

Secret key/value Plaintext

Secret Key	Secret Value
password	5S?0(78tE~t3Ve+!
dbname	demodb2
engine	mysql
port	3306
dbInstanceIdentifier	rotation-instance
host	rotation-instance.cbrfoqimfhmm.us-east-1.rds.amazonaws.com
username	\${Ref RDSUserName}

Example password stored in Secrets Manager

The Lambda function includes the following code to retrieve the secret from Secrets Manager. The function then uses it to connect to the database securely.

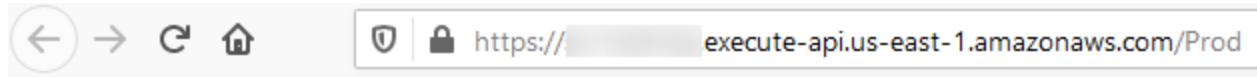
JavaScript

```
secret_name = os.environ['SECRET_NAME']
rds_host = os.environ['RDS_HOST']
name = os.environ['RDS_USERNAME']
db_name = os.environ['RDS_DB_NAME']

session = boto3.session.Session()
client = session.client(
    service_name='secretsmanager',
    region_name=region_name
)
get_secret_value_response = client.get_secret_value(
```

```
    SecretId=secret_name
)
...
secret = get_secret_value_response['SecretString']
j = json.loads(secret)
password = j['password']
...
```

Browsing to the endpoint URL specified in the CloudFormation output displays the number of records. This confirms that the Lambda function has successfully retrieved the secure database credentials and queried the table for the record count.



Selected 3 items from RDS MySQL table

Lambda function retrieving database credentials

Audit secrets access through a secrets manager

Monitor how your secrets are used to confirm that the usage is expected, and log any changes to them. This helps ensure that any unexpected usage or change can be investigated, and unwanted changes can be rolled back.

Hashicorp Vault uses [Audit devices](#) that keep a detailed log of all requests and responses to Vault. Audit devices can append logs to a file, write to syslog, or write to a socket.

Secrets Manager supports logging API calls with [AWS CloudTrail](#). CloudTrail captures all API calls for Secrets Manager as events. This includes calls from the Secrets Manager console and from code calling the Secrets Manager APIs.

Viewing the CloudTrail event history shows the requests to `secretsmanager.amazonaws.com`. This shows the requests from the console in addition to the Lambda function.

Event history (50+) Info

Download events

Create Athena table

Event history shows you the last 90 days of management events.

Event source

secretsmanager.amazonaws.com

30m1h3h12hCustom

< 1 2 ... >

<input type="checkbox"/>	Event name	Event time	User name	Event source	Resource type	Resource name
<input type="checkbox"/>	GetSecretValue	July 02, 2021, 11:58...		secretsmanager.amazonaws.com	AWS::SecretsManager::Secret	MyRDSInstanceRotationSecret-PkhGUveWw39z
<input type="checkbox"/>	GetSecretValue	July 02, 2021, 11:52...	demo-ssm-LambdaRDSTest-JA4QrUG05mn	secretsmanager.amazonaws.com	AWS::SecretsManager::Secret	arn:aws:secretsmanager:us-east-1:...
<input type="checkbox"/>	GetSecretValue	July 02, 2021, 11:48...		secretsmanager.amazonaws.com	AWS::SecretsManager::Secret	MyRDSInstanceRotationSecret-PkhGUveWw39z
<input type="checkbox"/>	DescribeSecret	July 02, 2021, 11:47...		secretsmanager.amazonaws.com	AWS::SecretsManager::Secret	MyRDSInstanceRotationSecret-PkhGUveWw39z
<input type="checkbox"/>	ListSecrets	July 02, 2021, 11:38...		secretsmanager.amazonaws.com	-	-
<input type="checkbox"/>	ListSecrets	July 02, 2021, 11:38...		secretsmanager.amazonaws.com	-	-
<input type="checkbox"/>	GetSecretValue	July 02, 2021, 11:37...	demo-ssm-LambdaRDSTest-JA4QrUG05mn	secretsmanager.amazonaws.com	AWS::SecretsManager::Secret	arn:aws:secretsmanager:us-east-1:...
<input type="checkbox"/>	GetSecretValue	July 02, 2021, 11:37...	demo-ssm-LambdaRDSTest-JA4QrUG05mn	secretsmanager.amazonaws.com	AWS::SecretsManager::Secret	arn:aws:secretsmanager:us-east-1:...

CloudTrail showing access to Secrets Manager

Secrets Manager also works with [Amazon EventBridge](#) so you can trigger alerts when administrator-specified operations occur. You can configure EventBridge rules to alert on deleted secrets or secret rotation. You can also create an alert if anyone tries to use a secret version while it is pending deletion. This can identify and alert when there is an attempt to use an out-of-date secret.

Enforce least privilege access to secrets

Access to secrets must be tightly controlled because the secrets contain sensitive information. Create [AWS Identity and Access Management](#) (IAM) policies that enable minimal access to secrets to prevent credentials being accidentally used or compromised. Secrets that have policies that are too permissive could be misused by other environments or developers. This can lead to accidental data loss or compromised systems. For more information, see [“Authentication and access control for AWS Secrets Manager”](#).

Rotate secrets frequently.

Rotating your workload secrets is important. This prevents misuse of your secrets since they become invalid within a configured time period.

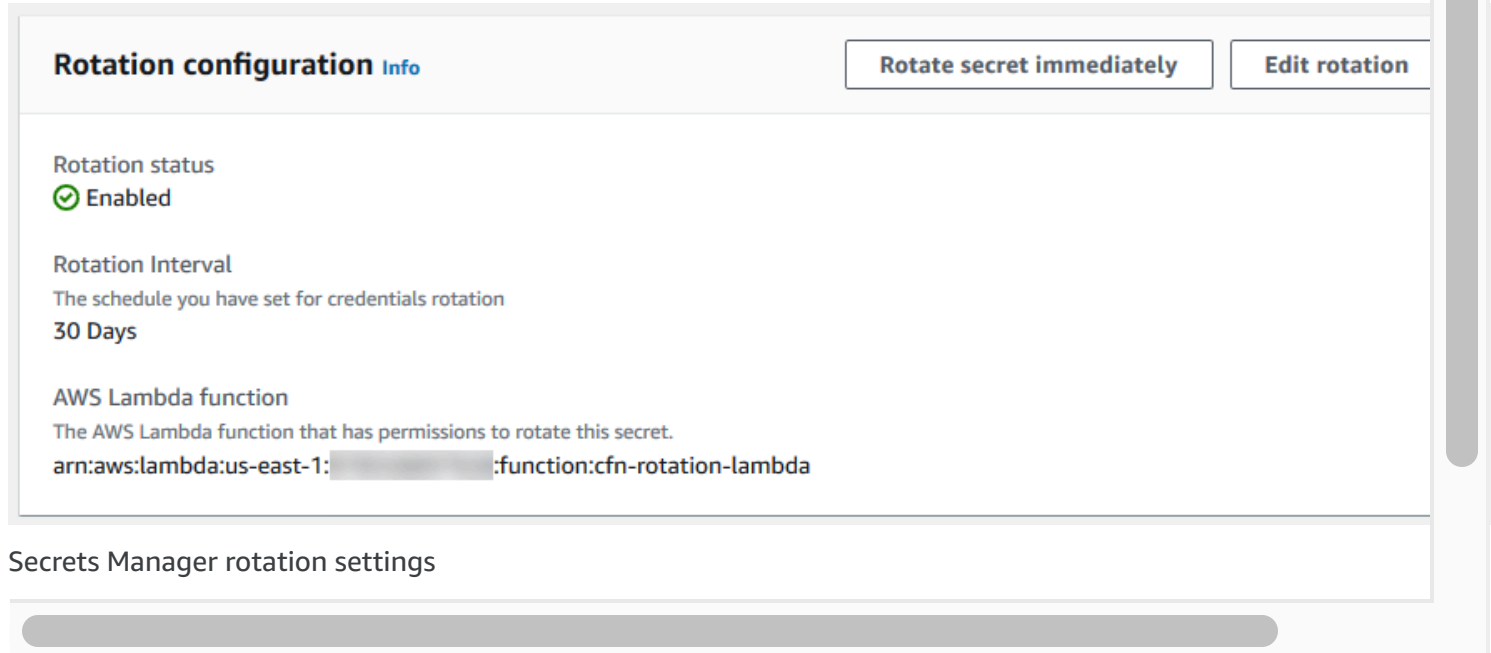
Secrets Manager allows you to rotate secrets on a schedule or on demand. This enables you to replace long-term secrets with short-term ones, significantly reducing the risk of compromise. Secrets Manager creates a CloudFormation stack with a Lambda function to manage the rotation process for you. Secrets Manager has native integrations with [Amazon RDS](#), [Amazon Redshift](#), and [Amazon DocumentDB](#). It populates the function with the Amazon Resource Name (ARN) of the secret. You specify the permissions to rotate the credentials, and how often you want to rotate the secret.

The CloudFormation stack creates a `MySecretRotationSchedule` resource with a `MyRotationLambda` function to rotate the secret every 30 days.

YAML

```
MySecretRotationSchedule:
  Type: AWS::SecretsManager::RotationSchedule
  DependsOn: SecretRDSInstanceAttachment
  Properties:
    SecretId: !Ref MyRDSInstanceRotationSecret
    RotationLambdaARN: !GetAtt MyRotationLambda.Arn
    RotationRules:
      AutomaticallyAfterDays: 30
MyRotationLambda:
  Type: AWS::Serverless::Function
  Properties:
    Runtime: python3.7
    Role: !GetAtt MyLambdaExecutionRole.Arn
    Handler: mysql_secret_rotation.lambda_handler
    Description: 'This is a lambda to rotate MySQL user passwd'
    FunctionName: 'cfn-rotation-lambda'
    CodeUri: 's3://devsecopsblog/code.zip'
```

View and edit the rotation settings in the Secrets Manager console.



Rotation configuration [Info](#)

[Rotate secret immediately](#) [Edit rotation](#)

Rotation status
✔ Enabled

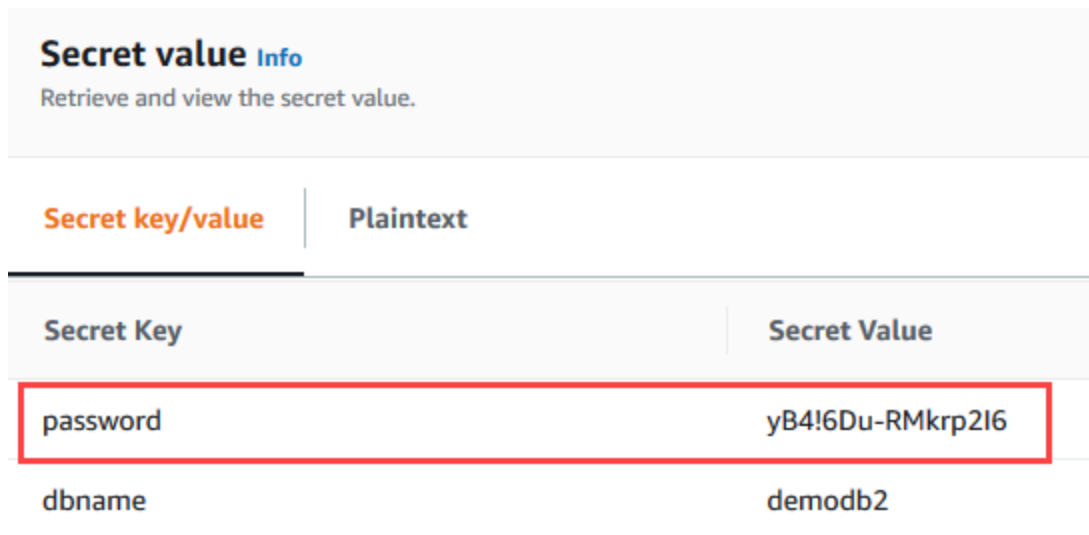
Rotation Interval
The schedule you have set for credentials rotation
30 Days

AWS Lambda function
The AWS Lambda function that has permissions to rotate this secret.
arn:aws:lambda:us-east-1:[redacted]:function:cfn-rotation-lambda

Secrets Manager rotation settings

the database password and updates the secret in Secrets Manager.

View the updated secret in Secrets Manager, where the password has changed.



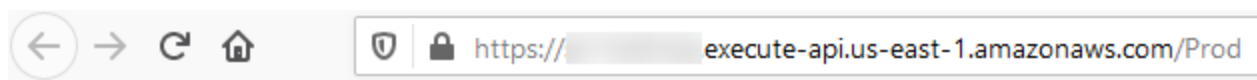
Secret value [Info](#)
Retrieve and view the secret value.

Secret key/value | Plaintext

Secret Key	Secret Value
password	yB4!6Du-RMkrp2I6
dbname	demodb2

Secrets Manager password change

Browse to the endpoint URL to confirm you can still access the database with the updated credentials.



Selected 3 items from RDS MySQL table

Access endpoint with updated Secret Manager password

You can provide your own code to customize a Lambda rotation function for other databases or services. The code includes the commands required to interact with your secured service to update or add credentials.

Conclusion

Implementing application security in your workload involves reviewing and automating security practices at the application code level. By implementing code security, you can protect against emerging security threats. You can improve the security posture by checking for malicious code, including third-party dependencies.

In this post, I continue from [part 1](#), looking at securely storing, auditing, and rotating secrets that are used in your application code.

In the next post in the series, I start to cover the reliability pillar from the Well-Architected Serverless Lens with regulating inbound request rates.

For more serverless learning resources, visit [Serverless Land](#).

TAGS: [serverless](#), [well-architected](#)