International Journal of
Technology and Systems
(IJTS)

Strategic Implementation of AWS Security Services: A Focus on Best Practices, SSM and Secrets Manager

CARI Journals

Journal of Technology and Systems
ISSN : 2788-6344 (Online)
Vol. 6, Issue No. 1, pp 25 – 35, 2024
www.carijournals.org

Strategic Implementation of AWS Security Services: A Focus on Best Practices, SSM and Secrets Manager

Balasubrahmanya Balakrishna

Richmond, VA, USA

https://orcid.org/0009-0000-1195-123X

Accepted: 7th Jan 2024 Received in Revised Form: 21st Jan 2024 Published: 4th Feb 2024

Abstract

Purpose: As organizations increasingly migrate to the cloud, ensuring robust data security becomes paramount. This technical paper explores the purpose, methodology, findings, and unique contributions in cloud security, focusing on AWS Systems Manager (SSM) and Secrets Manager. The objective is to provide a comprehensive guide for engineers and architects seeking to enhance data security in their cloud environments.

Methodology: The comparative study conducted involves analyzing the features, capabilities, and integration possibilities of AWS SSM and Secrets Manager. Real-world use cases, best practices, and security measures associated with these services are explored through a methodology that includes hands-on exploration, case studies, and a survey of existing literature to distill critical insights.

Findings: The paper uncovers a rich landscape of security measures facilitated by AWS SSM and Secrets Manager. Findings highlight the strengths and limitations of each service, emphasizing the importance of their integration for a holistic security approach. Automated rotation of credentials, encryption options, IAM policies, and monitoring strategies emerge as critical findings, contributing to the overall understanding of secure cloud data management.

Unique contributor to theory, policy, and practice: This work uniquely provides a detailed comparative analysis that empowers engineers and architects to make well-informed decisions. The paper offers insights into the nuances of AWS SSM and Secrets Manager, enabling professionals to tailor their security strategies based on specific use cases and requirements. The focus on real-world scenarios and identifying best practices make this contribution practical and applicable, serving as a valuable resource for those navigating the complex landscape of cloud security engineering and architecture.

Keywords: *AWS SSM, AWS Secret Manager, Data Security, KMS secrets encryption, and decryption*

INTRODUCTION

BACKGROUND

As a fundamental practice, application secrets should never be stored in plain text within environment variables in AWS Lambda. In many organizations, these secrets undergo encryption using a Key Management Service (KMS) key and are then securely stored in the AWS SSM Parameter Store or Secrets Manager. This ensures that the secrets remain protected at rest and during access through Identity and Access Management (IAM) policies[1].

However, a challenge arises during deployment. Environment variables reference encrypted and stored securely Parameters in configuration files such as serverless.yml in the Serverless framework. Unfortunately, during this process, the parameters are decrypted and exposed as plain text in environment variables for Lambda functions. While this setup facilitates easy access to secrets within the code, it simultaneously introduces a potential vulnerability, as it becomes susceptible to exploitation by malicious actors. The depicted pattern is illustrated in Figure 1 below.

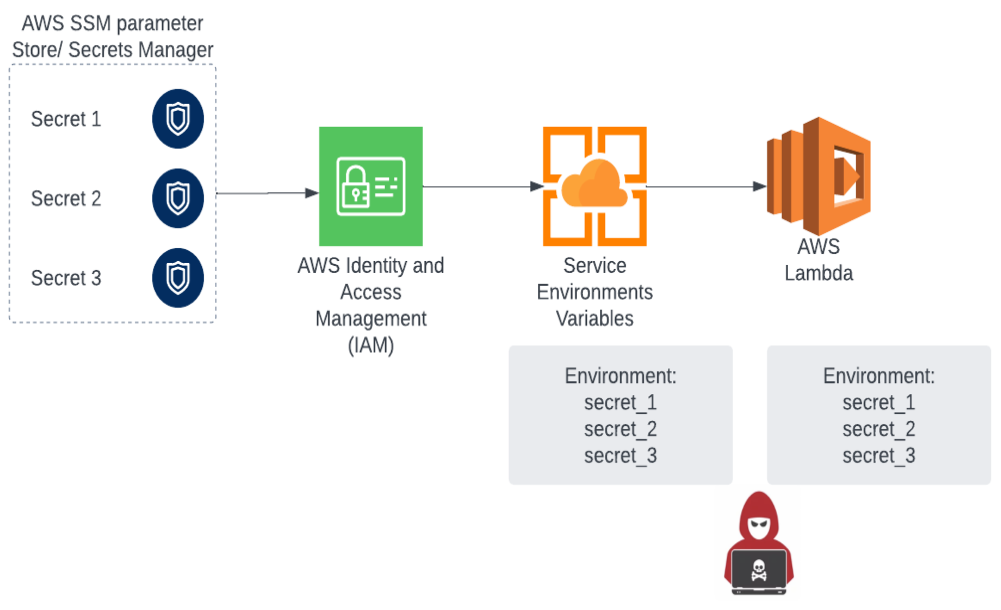

AWS SSM parameter Store/ Secrets Manager
Secret 1
Secret 2
Secret 3
AWS Identity and Access Management (IAM)
Service Environments Variables
AWS Lambda
Environment: secret_1 secret_2 secret_3
Environment: secret_1 secret_2 secret_3

Figure 1. AWS Secrets in Environment Variable

The recommended strategy, illustrated in Figure 2, entails fetching and decrypting parameters dynamically during the initial startup, commonly known as the cold start phase. In this approach, the function requires knowledge of the parameter's name, which can still be passed through environment variables. To alleviate the necessity of querying the Secrets Manager or SSM

Parameter Store for every request, it is prudent to implement an effective caching mechanism, periodically refreshing the cached secrets at defined intervals. This approach strikes a balance between accessing required secrets and minimizing data retrieval frequency, thereby optimizing both performance and security.

Furthermore, this method enables effortless secret rotation without needing to redeploy the Lambda function that relies on these secrets. This guarantees the dynamism and security of the system, permitting updates to secrets without causing disruptions to the ongoing operations of the associated Lambda functions. AWS Powertools provides essential libraries and functions, streamlining the efficient execution of these operations [2].

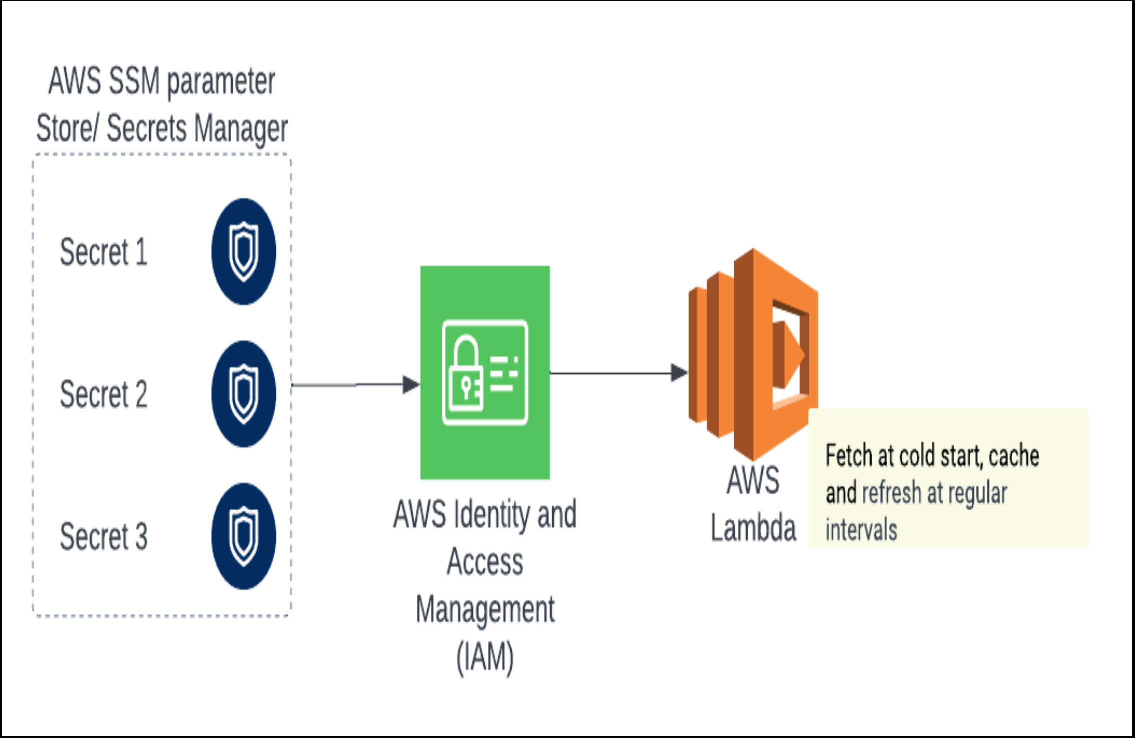

AWS SSM parameter Store/ Secrets Manager
Secret 1
Secret 2
Secret 3
AWS Identity and Access Management (IAM)
AWS Lambda
Fetch at cold start, cache and refresh at regular intervals

Figure 2. AWS Secrets fetched during Cold Start

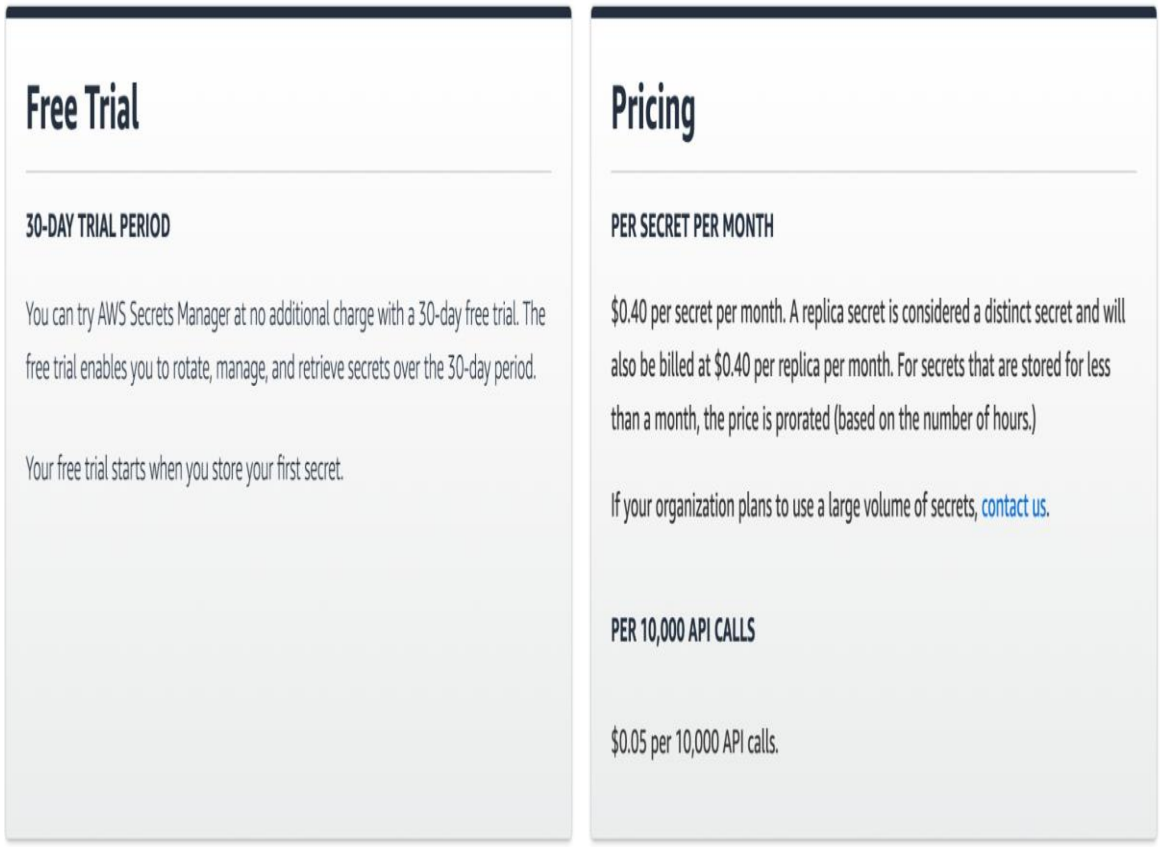
DECODING AWS SECRET STORAGE: SSM PARAMETER STORE VS. SECRETS MANAGER

As highlighted before, two primary options emerge in addressing the imperative security aspect of storing secrets on AWS – AWS Systems Manager (SSM) Parameter Store and AWS Secrets Manager. Irrespective of the chosen secret storage service, specific foundational requirements must be met to ensure robust security:

1. *Encryption Standards*: Implementation of encryption protocols for both data at rest and during transit. Encryption guarantees that secrets remain secure, whether stored or in transit, aligning with industry-leading security practices.

2. *Role-Based Access Control (RBAC)*: Enabling role-based access control through IAM (Identity and Access Management), ensuring fine-grained control over who can access specific secrets. This approach enhances security by limiting access to authorized entities only.

3. *Cost-Efficiency Model:* Adopting a cost-efficient model that follows a pay-per-usage structure rather than being tied to uptime. An efficient model ensures optimized expenditure while maintaining the flexibility to scale resources based on actual utilization.



Free Trial

30-DAY TRIAL PERIOD

You can try AWS Secrets Manager at no additional charge with a 30-day free trial. The free trial enables you to rotate, manage, and retrieve secrets over the 30-day period.

Your free trial starts when you store your first secret.

Pricing

PER SECRET PER MONTH

$0.40 per secret per month. A replica secret is considered a distinct secret and will also be billed at $0.40 per replica per month. For secrets that are stored for less than a month, the price is prorated (based on the number of hours.)

If your organization plans to use a large volume of secrets, contact us.

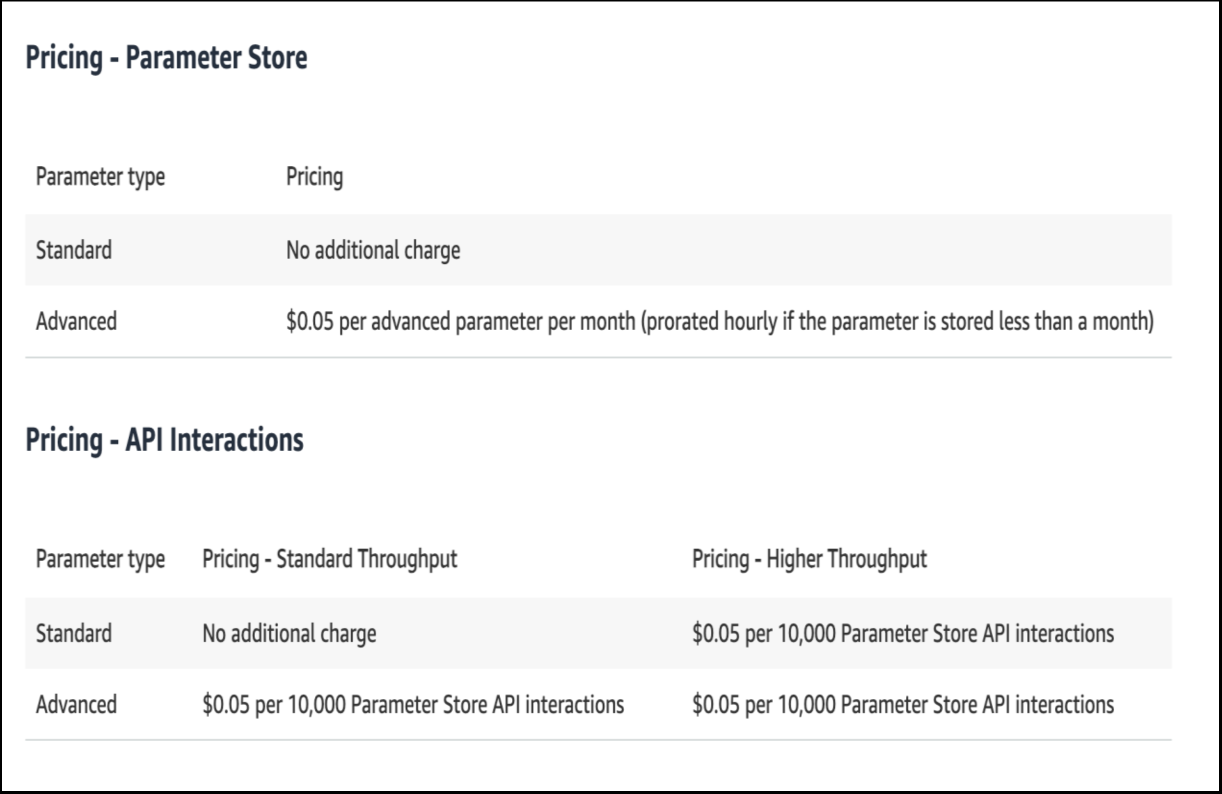
PER 10,000 API CALLS

$0.05 per 10,000 API calls.

Figure 2. AWS Secrets Manager Pricing

As observed in Figure 2, Secret Manager Services imposes a charge of $0.40 per month and an additional $0.05 per 10,000 calls. The cost the implication of 40 cents per call can swiftly escalate, mainly when operating at scale [3]. Notably, users are still subject to charges even if the service is not actively utilized for storing or retrieving secrets. This aspect introduces a potential cost burden, emphasizing the importance of strategic cost considerations when evaluating and using the service.

Journal of Technology and Systems
ISSN : 2788-6344 (Online)
Vol. 6, Issue No. 1, pp 25 – 35, 2024
www.carijournals.org



Pricing - Parameter Store

Parameter type	Pricing
Standard	No additional charge
Advanced	$0.05 per advanced parameter per month (prorated hourly if the parameter is stored less than a month)

Pricing - API Interactions

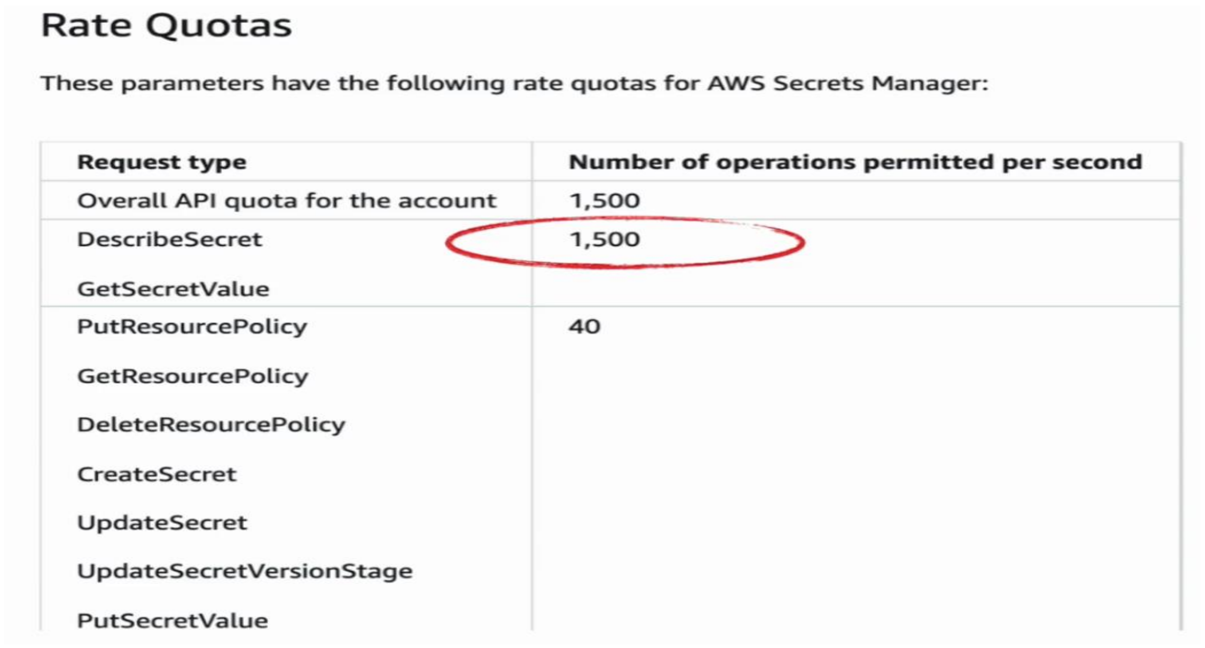
Parameter type	Pricing - Standard Throughput	Pricing - Higher Throughput
Standard	No additional charge	$0.05 per 10,000 Parameter Store API interactions
Advanced	$0.05 per 10,000 Parameter Store API interactions	$0.05 per 10,000 Parameter Store API interactions

Figure 3. AWS SSM Parameter Store Pricing

Contrastingly, as shown in Figure 3, the SSM Parameter Store stands out as a no-cost service for standard throughput. Users can access advanced parameters for a mere 5 cents per month, offering a larger payload limit and the ability to control aspirations and change notifications [4]. While the standard parameter type is generally sufficient for most use cases, opting for the advanced parameter type becomes ideal in a production environment that demands higher throughput. However, knowing that the SSM Parameter Store transitions from a free service to charging 5 cents per 10,000 requests[5] upon enabling the advanced parameter type is crucial.

Regarding cost efficiency, Secrets Manager exhibits a rapid cost escalation, mainly when an application extensively utilizes secrets. Notably, it's essential to recognize that additional costs may be incurred for Key Management Service (KMS) usage, applicable to SSM Parameter Store and Secrets Manager for the encryption and decryption of secrets. Therefore, a meticulous evaluation of cost implications, considering service usage and associated dependencies like KMS, is imperative for informed decision-making.

SSM Parameter Store can store secrets as plain text or string with encryption. On the other hand, Secrets Manager gives a reasonably high API rate limit of 1500 secrets per second [6], as shown in Figure 4.

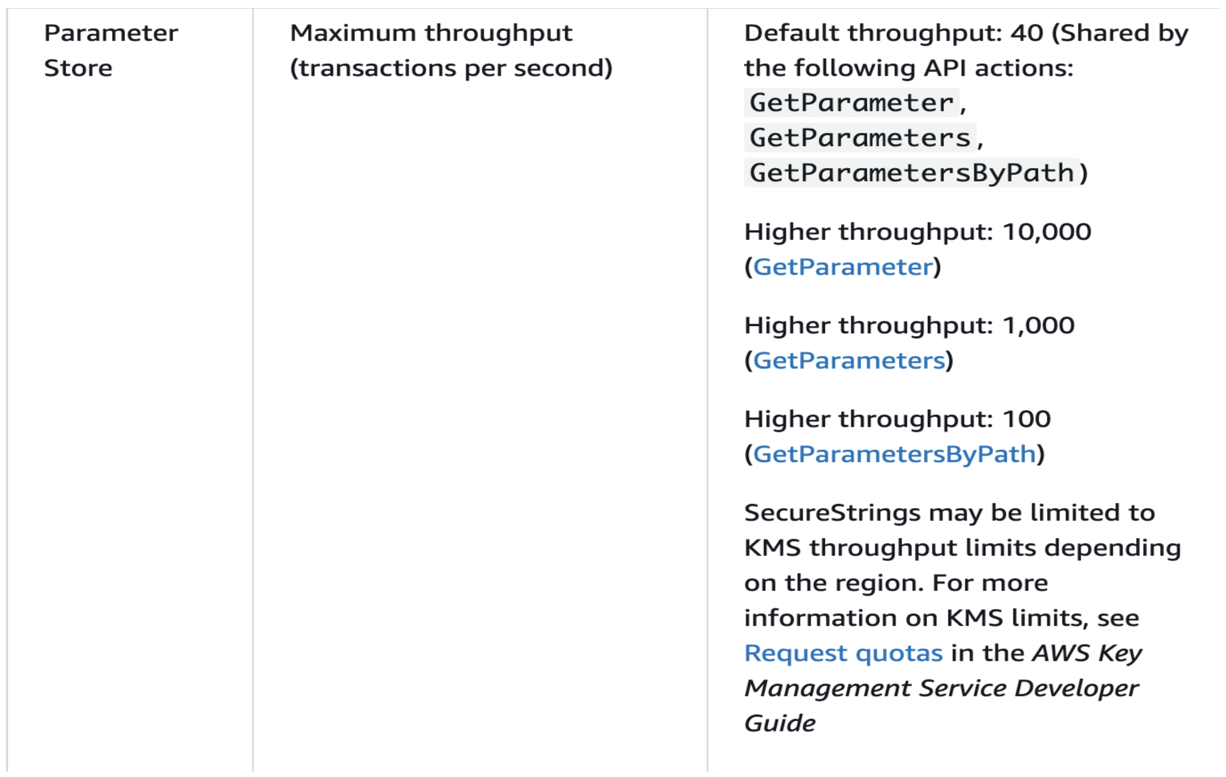

Rate Quotas

These parameters have the following rate quotas for AWS Secrets Manager:

Request type	Number of operations permitted per second
Overall API quota for the account	1,500
DescribeSecret GetSecretValue	1,500
PutResourcePolicy GetResourcePolicy DeleteResourcePolicy CreateSecret UpdateSecret UpdateSecretVersionStage PutSecretValue	40

Figure 4. AWS Secret Manager Rate Quotas

Conversely, the SSM Parameter Store provides merely 40 secrets per second throughput, as shown in Figure 5. So, it is advised to enable advanced higher throughputs, which would be rate-limited at 10,000 secret requests per second.



Parameter Store	Maximum throughput (transactions per second)	Default throughput: 40 (Shared by the following API actions: `GetParameter`, `GetParameters`, `GetParametersByPath`) Higher throughput: 10,000 (GetParameter) Higher throughput: 1,000 (GetParameters) Higher throughput: 100 (GetParametersByPath) SecureStrings may be limited to KMS throughput limits depending on the region. For more information on KMS limits, see Request quotas in the *AWS Key Management Service Developer Guide*

Figure 5. SSM Parameter Store Throughput

4. *Scalability with Lambda Functions:* Seamless integration with AWS Lambda functions allows scalability and dynamic resource allocation. Scalability ensures the secret storage service can adapt to varying workloads and demands effortlessly.

5. *Full Management by AWS:* Full management and maintenance by AWS alleviates the burden of operational overhead from users. Fully managed service ensures the latest security updates and patches and provides a reliable and consistent service.

By adhering to these fundamental requirements, AWS SSM Parameter Store and AWS Secrets Manager offer a secure foundation for storing secrets in the cloud. This comprehensive approach addresses encryption, access control, cost-effectiveness, scalability, and management simplicity, laying the groundwork for a robust and resilient secret management system on the AWS platform.

Comparative Analysis: Built-In Key Rotation in AWS SSM Parameter Store and Secrets Manager

Consideration should be given to the integrated key rotation feature when assessing AWS secret management services. While AWS Secrets Manager and AWS Systems Manager (SSM) Parameter Store have different features, they provide essential rotation functions.

A. AWS SSM Parameter Store

Manual Key Rotation:

- SSM Parameter Store provides flexibility by allowing manual key rotation. Users have the option to trigger key rotation at their discretion.

Customization and Control:

- While key rotation is not automated, this manual approach offers users greater customization and control over the rotation process. Organizations can tailor the rotation to align with specific security policies and compliance requirements with a simple cron job to rotate the secrets easily.

B. AWS Secrets Manager

Automated Key Rotation:

- Secrets Manager takes a more automated approach to key rotation. It offers automatic, scheduled key rotation for supported services, such as Amazon RDS database credentials.

Ease of Use:

- This automated mechanism simplifies the key rotation process, reducing the operational overhead on users. It is particularly advantageous for scenarios where regular rotation is crucial for maintaining security.

DECODING AWS SECRETS MANAGER: FEATURES THAT SET IT APART

AWS Secrets Manager boasts highly beneficial features for streamlined and secure secret management in an enterprise environment.

A. Resource Policy for Centralized Control:

Secrets Manager introduces the concept of Resource Policy, empowering organizations to centralize all secrets in an account controlled by information security or platform teams. This centralized control allows for finely tuned access permissions, facilitating comprehensive governance from a single, authoritative source.

B. Regional Replication for Multi-Region Setup:

With AWS Secrets Manager, the capability to replicate secrets across multiple regions is a notable advantage. In a multi-region setup, this feature ensures consistency and availability. While there is a cost implication due to extra copies of secrets, the convenience and reliability of regional replication justify the investment compared to the manual copying process required with the SSM Parameter Store. This extra cost is a small price for the enhanced security and resilience achieved through regional redundancy.

Rationale And Conclusion

The choice of relevant data security tools is crucial in the dynamic world of cloud computing. To protect sensitive data in the cloud, this paper thoroughly justifies selecting AWS Systems Manager (SSM) Parameter Store and AWS Secrets Manager.

Robust secret management solutions are imperative given the increasing popularity of cloud services, and AWS provides two well-liked options: SSM Parameter Store and Secrets Manager. For engineers and architects to make well-informed judgments, it is essential to comprehend the reasoning behind their features and functionalities.

The comparison analysis clarifies the subtle differences between these services. AWS SSM Parameter Store touts itself as a configurable solution with its manual key rotation and granular management, making it desirable to enterprises with specific security policies and compliance requirements. Conversely, AWS Secrets Manager prioritizes regularity and simplicity with its automated key rotation feature, making it an excellent option for environments that value user-friendliness above strict security protocols.

Examining the unique characteristics of every service becomes essential for customers looking for low-cost options. Although SSM Parameter Store is reasonably priced for average traffic, AWS Secrets Manager has fees that can add up quickly, mainly when apps often use secrets. Furthermore, the Key Management Service (KMS), which applies to secret encryption and decryption services, is included in the cost considerations.

The paper delves into the details of AWS Secrets Manager, emphasizing its features for regional replication and resource policy for centralized control. The Resource Policy provides unmatched control and access management by allowing the consolidation of secrets under centralized governance. Regional replication surpasses the SSM Parameter Store's manual copying procedure, even if it comes at an extra cost, and offers the crucial benefit of redundancy in a multi-region configuration.

In conclusion, exploring AWS Systems Manager (SSM) Parameter Store and AWS Secrets Manager unravels critical dimensions for architects and engineers in cloud security. This comparative analysis serves as a compass, guiding decision-makers through the intricate landscape of data protection and secret management.

This comprehensive analysis should provide decision-makers with all necessary parameters to consider when deciding between SSM and Secrets Manager adoption. It equips professionals with the insights needed to align their choices with organizational priorities, compliance requirements, and operational needs in the dynamic landscape of cloud security.

Recommendation

This paper's thorough study indicates that individual corporate priorities, compliance concerns, and operational requirements will determine which AWS Systems Manager (SSM) Parameter

Store and AWS Secrets Manager are best. Consider the following recommendations for decision-makers:

A. *Choose AWS SSM Parameter Store if:*

- Specific security policies and compliance requirements necessitate granular management and manual key rotation.
- Cost considerations are crucial, and applications experience average traffic, as SSM Parameter Store is reasonably priced for such scenarios.

B. *Opt for AWS Secrets Manager if:*

- User-friendliness and simplicity are prioritized, and automated key rotation is a critical requirement.
- Applications frequently use secrets, and cost is not the primary concern, as AWS Secrets Manager fees may accumulate with frequent secret usage.
- Regional replication and centralized control through Resource Policy are essential for security and access management strategy.

C. *Considerations for Both:*

- Evaluate Key Management Service's (KMS) cost implications for secret encryption and decryption, as it contributes to overall expenses.
- Assess redundancy needs in a multi-region configuration, as AWS Secrets Manager's regional replication provides a crucial benefit in this context.

D. *Final Considerations:*

- Align choices with organizational priorities, compliance requirements, and operational needs to ensure seamless integration of the selected solution into the cloud security architecture.
- Regularly reassess organizational evolving needs to ensure the selected solution remains aligned with changing circumstances.

References

[1] AWS (2022, May 8). *Securely retrieving secrets with AWS Lambda*. AWS Compute Blog. https://aws.amazon.com/blogs/compute/securely-retrieving-secrets-with-aws-lambda/

[2]AWS (n.d.). Parameters. AWS Powertools for AWS Lambda (Python). https://docs.powertools.aws.dev/lambda/python/latest/utilities/parameters/

[3]AWS (n.d.). *AWS Secrets Manager Pricing*. AWS Secrets Manager. https://aws.amazon.com/secrets-manager/pricing/

[4]AWS (n.d.). *Parameter Store*. AWS Systems Manager Pricing. https://aws.amazon.com/systems-manager/pricing/

[5]AWS (2022, November 11). *AWS Secrets Manager increases the API Requests per Second limits*. What's New With AWS? https://aws.amazon.com/about-aws/whats-new/2023/07/aws-systems-manager-parameter-store-api-limit/

[6]AWS (n.d.). *AWS Secrets Manager quotas*. AWS Secrets Manager. https://docs.aws.amazon.com/secretsmanager/latest/userguide/reference_limits.html