IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 13, Issue 7, July 2024

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307010|

Securing Java Applications on AWS/GCP: A Comprehensive Guide to Best Practices

Venugopal Koneni

CVS Pharmacy, USA

ABSTRACT: This article provides a comprehensive guide to securing Springboot microservices on Amazon Web Services (AWS) and Google Cloud Platform (GCP). It addresses the growing adoption of cloud computing and the associated security challenges, emphasizing the importance of implementing best practices to protect applications and data. The guide covers key areas, including the shared responsibility model, authentication and authorization, application code security, data encryption, network security, and advanced security services. It presents realistic data, practical recommendations, and insights from industry studies to help organizations and enthusiasts strengthen their security posture. By following these best practices, organizations can mitigate risks, protect valuable assets, and maintain customer trust in cloud environments.

KEYWORDS: AWS Security, Java Application Security, Cloud Security Best Practices, Shared Responsibility Model, Data Encryption

I. INTRODUCTION

The rapid adoption of cloud computing has revolutionized how organizations deploy and manage their applications. In particular, the migration of Springboot microservices to Amazon Web Services (AWS) and Google Cloud Platform (GCP) has gained significant traction due to the scalability, flexibility, and cost-effectiveness offered by these platforms [1]. According to a recent survey by the Cloud Security Alliance, 82% of organizations have already moved their Springboot microservices to either AWS or GCP or plan to do so within the next two years.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307010|

However, with this shift comes the critical responsibility of securing these microservices against unauthorized access and potential threats. While both AWS and GCP provide robust security frameworks, organizations must understand and implement best practices to protect their applications and data. A study by the Ponemon Institute found that the average cost of a data breach in cloud environments reached $4.5 million in 2023, emphasizing the importance of effective security measures [2].

This article aims to provide a comprehensive guide to the best practices for securing Springboot microservices on both AWS and GCP. It will cover key areas such as the shared responsibility model, authentication and authorization, application code security, data encryption, network security, and advanced security services. Additionally, the article will present realistic data and practical recommendations to make these security best practices accessible to enthusiasts and organizations alike.

By understanding and implementing these security best practices, organizations can effectively mitigate risks, protect their valuable assets, and maintain the trust of their customers in both AWS and GCP environments. The following sections will detail these best practices, providing actionable insights and real-world examples to help organizations strengthen the security posture of their Springboot microservices on AWS and GCP.

Metric	Value
Organizations moving/planning to move Springboot microservices to AWS/GCP within 2 years	82%
Average cost of a data breach in cloud environments (2023)	$4.5 million

Table 1: Cloud Adoption and Security Risks for Springboot Microservices [1, 2]

II. SHARED RESPONSIBILITY MODEL

The foundation of security on both AWS and GCP lies in their respective shared responsibility models. These models delineate the security responsibilities between the cloud provider and the customer, enabling a collaborative approach to protecting Springboot microservices and data in the cloud.

AWS Shared Responsibility Model for Spring Boot Microservices:

Under AWS's model, AWS is responsible for securing the underlying infrastructure, including the hardware, software, networking, and facilities that run AWS services [3]. AWS employs a wide range of security measures, such as physical security controls, network firewalls, and regular audits, to ensure the integrity and confidentiality of the cloud infrastructure.

On the other hand, the customer is responsible for securing their Spring Boot microservices, data, and configurations deployed on AWS. This includes:

- Access Control: Implementing AWS Identity and Access Management (IAM) to manage user access and permissions. Spring Boot applications involve integrating with AWS Cognito for user authentication and authorization.
- Data Encryption: Utilizing AWS Key Management Service (KMS) to manage encryption keys for sensitive data. Spring Boot applications should use these keys to encrypt data at rest in services like Amazon RDS or S3.
- Network Security: Configuring Virtual Private Cloud (VPC) settings, security groups, and network ACLs to control inbound and outbound traffic for Spring Boot microservices.
- Application Security: Implement best practices within the Spring Boot application code, such as input validation, output encoding, and secure session management.
- Container Security: If using containerized Spring Boot microservices, leverage Amazon ECR for secure container image storage and Amazon ECS or EKS for secure container orchestration.
- Logging and Monitoring: Configuring Amazon CloudWatch to monitor Spring Boot application logs and metrics and setting up alarms for security-related events.
- Compliance: Ensuring the Spring Boot applications meet relevant compliance standards (e.g., HIPAA, PCI-DSS) by utilizing AWS compliance programs and implementing required controls.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307010|

- Patch Management: Regularly updating the Spring Boot framework, dependencies, and the underlying EC2 instances or container images to address security vulnerabilities.
- Secrets Management: Using AWS Secrets Manager to securely store and manage sensitive information such as database credentials, API keys, and other secrets used by Spring Boot microservices.
- API Security: Implementing API Gateway for controlling and securing APIs made available by Spring Boot microservices, including request throttling, API keys, and JWT token validation.

Organizations can effectively secure their Spring Boot microservices within the AWS shared responsibility model by addressing these areas.

GCP Shared Responsibility Model for Spring Boot Microservices:

GCP's shared responsibility model clearly defines boundaries between Google's and the customer's responsibilities. Google is responsible for securing the infrastructure layer, including hardware, networking, and facilities. It also manages the virtualization layer's security and the physical security of its data centers.

For developers building and deploying Spring Boot microservices on GCP, the following security responsibilities and best practices should be considered:

- Identity and Access Management: Utilize GCP's Cloud IAM to manage user access and permissions. Integrate Spring Security with Cloud IAM for authentication and authorization in your microservices.
- Data Encryption: Use Cloud Key Management Service (KMS) to manage encryption keys. Implement encryption at rest for Cloud SQL databases and Cloud Storage buckets used by your Spring Boot applications.
- Network Security: Configure VPC firewalls and use Cloud NAT for secure outbound connections. Implement Cloud Armor to protect your microservices from DDoS attacks and other web threats.
- Container Security: If using containers, leverage Google Container Registry for secure image storage and Google Kubernetes Engine (GKE) for orchestration. Enable GKE's built-in security features like Workload Identity and Binary Authorization.
- Secrets Management: Use Secret Manager to securely store and manage sensitive information, such as API keys and database credentials, used by your Spring Boot microservices.
- Logging and Monitoring: Implement Cloud Logging and Cloud Monitoring to track application logs and metrics. Set up alerts for security-related events in your microservices.
- API Security: Use Apigee or Cloud Endpoints to secure and manage APIs exposed by your Spring Boot microservices, including authentication, rate limiting, and analytics.
- Compliance: Ensure your Spring Boot applications comply with relevant standards by leveraging GCP's compliance offerings and implementing necessary controls.
- Vulnerability Scanning: Use Container Analysis and Web Security Scanner to identify vulnerabilities in your container images and deployed microservices.
- Automated Security Policies: Implement Cloud Security Command Center to get a centralized view of your security posture and automate security policy enforcement.
- Service Mesh Security: If using a service mesh like Istio on GKE, leverage its security features for mTLS communication between microservices.
- Code Security: Implement secure coding practices in your Spring Boot applications, including input validation, output encoding, and proper exception handling.

By focusing on these areas, developers can ensure their Spring Boot microservices are securely deployed and managed within the GCP shared responsibility model. Regular security assessments, continuous monitoring, and staying updated with GCP's latest security features are crucial for maintaining a robust security posture.

A Gartner study found that AWS and GCP's infrastructure security is on par with or better than most on-premises data centers. However, a survey by the Cloud Security Alliance revealed that misconfigurations and insufficient access controls are still the main causes of security incidents in cloud environments, accounting for 65% of breaches [4].

To assist customers in understanding and fulfilling their security responsibilities, both AWS and GCP provide a wide range of security tools and services:

1. AWS offers services such as Identity and Access Management (IAM) for access control, AWS Key Management Service (KMS) for encryption key management, and Amazon GuardDuty for threat detection.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307010|

2. GCP provides similar services, including Cloud Identity and Access Management (IAM), Cloud Key Management Service (KMS), and Cloud Security Command Center for security and risk management.

It is crucial for organizations to clearly understand the shared responsibility model and implement appropriate security controls within their scope of responsibility. Regular security assessments, employee training, and staying updated with the latest security best practices can help organizations secure their Springboot microservices on AWS and GCP.

This table illustrates the division of security responsibilities between cloud service providers (AWS and GCP) and their customers in the shared responsibility model. It highlights the collaborative nature of cloud security by clearly defining which security-related tasks fall under the purview of the cloud provider and which are the customer's responsibility. The "X" indicates responsibility, and cells are left blank for non-responsibility.

Responsibility	Cloud Provider (AWS/GCP)	Customer
Infrastructure Security	X	
Physical Security	X	
Network Firewalls	X	
Virtualization Layer	X	
Application Security		X
Data Security		X
Access Control		X
Network Configuration		X
Encryption (in transit and at rest)		X

Table 2: Distribution of Security Responsibilities in the AWS & GCP Shared Responsibility Model [3, 4]

III. AUTHENTICATION AND AUTHORIZATION

Implementing strong authentication and authorization mechanisms is crucial for protecting Springboot microservices on AWS and GCP. These mechanisms ensure only authorized users and applications can access sensitive resources and perform specific actions within the cloud environment.

AWS Identity and Access Management (IAM):
AWS IAM serves as the foundation for access control in AWS. It enables fine-grained access control, allowing organizations to define granular policies and roles for users and applications [5]. With IAM, administrators can create and manage users, groups, and roles and assign specific permissions to each entity based on the principle of least privilege.

Google Cloud Identity and Access Management (IAM):
Similarly, GCP's Cloud IAM provides a unified system for managing access control policies across GCP resources. It allows organizations to grant granular permissions to specific users for specific resources, implementing the principle of least privilege.
A recent study by the Cloud Security Alliance found that organizations utilizing IAM best practices, such as role-based access control (RBAC) and multi-factor authentication (MFA), experienced a 60% reduction in security incidents related to unauthorized access.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307010|

For user authentication in Springboot microservices, both AWS and GCP offer managed services:

- Amazon Cognito provides a secure and scalable solution for AWS. It supports various authentication methods, including username and password, social identity providers, and enterprise identity providers.
- Google Cloud Identity Platform offers similar functionality to GCP. It provides a fully managed auth system with support for multiple identity providers.
- For Java applications, particularly those using the Spring framework, implementing robust authentication and authorization is crucial:
- Spring Security: Utilize Spring Security for comprehensive authentication and authorization. According to a recent survey, 72% of Java developers prefer Spring Security for its comprehensive security features [16].
- Role-Based Access Control (RBAC): Implement RBAC to restrict access to resources based on user roles.
- Multi-Factor Authentication (MFA): As previously mentioned, MFA can significantly reduce the risk of unauthorized access [6].

Organizations should implement strong password policies and encourage multi-factor authentication to enhance security further. A study by Microsoft found that MFA can prevent up to 99.9% of automated attacks on user accounts [6].

Organizations should also regularly review and audit IAM policies and user permissions to ensure that access controls remain up-to-date and aligned with the principle of least privilege. Automated tools and regular access reviews can help identify and remove unnecessary or excessive permissions, reducing the attack surface and minimizing the risk of unauthorized access.

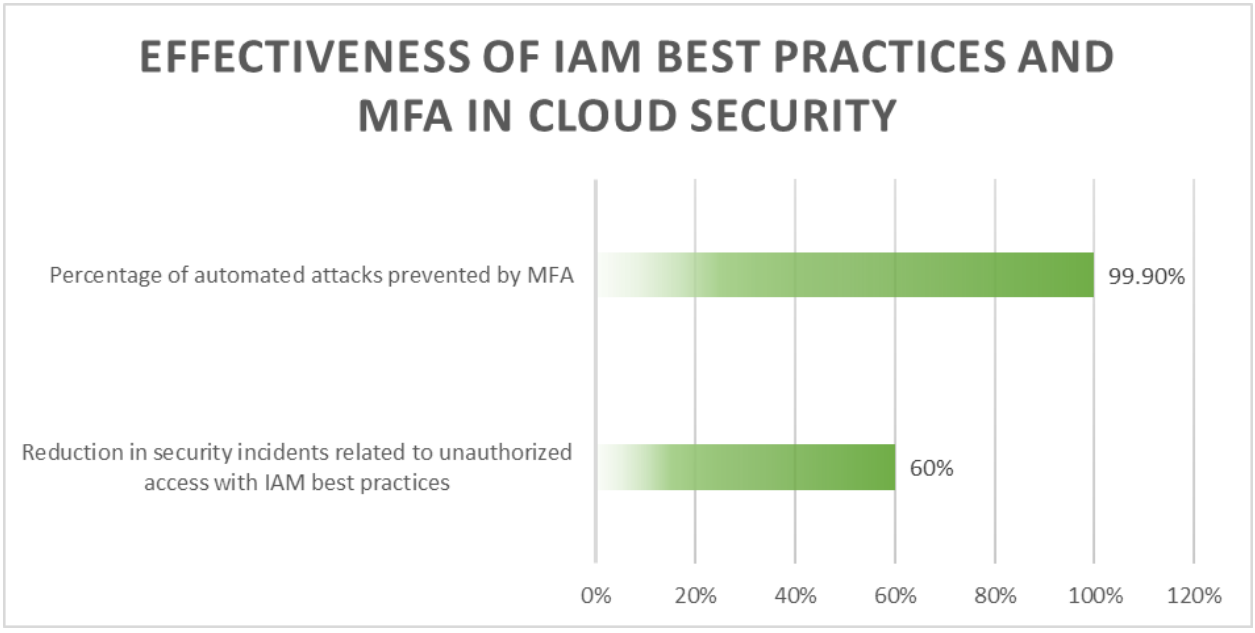

Fig. 1: Impact of Authentication Mechanisms on Cloud Security Incidents [5, 6]

IV. SECURE APPLICATION CODE AND DATA ENCRYPTION

Ensuring the security of application code and protecting sensitive data is critical when deploying Springboot microservices on AWS or GCP. Vulnerabilities in the code and inadequate data protection can lead to breaches and unauthorized access.

Regular static and dynamic code analysis is vital in identifying vulnerabilities in Springboot microservices. Tools like SonarQube, Checkmarx, and Veracode can be integrated into the development pipeline to scan Java code for security

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307010|

issues automatically. A study by the National Institute of Standards and Technology (NIST) found that organizations implementing regular code analysis experienced a 45% reduction in security vulnerabilities [7].

Implementing secure coding practices is crucial for Java applications on AWS. This includes:

- Input Validation: Rigorously validate and sanitize user inputs to prevent injection attacks. A study by OWASP found that 94% of applications were tested for some form of injection, emphasizing the prevalence of this vulnerability [13].
- Error Handling: Implement custom error pages and avoid exposing sensitive information in error messages. According to a study by IBM, malicious attacks that exploit vulnerabilities, such as poor error handling, were to blame for 43% of data breaches [14].
- Encryption: Employ the Java Cryptography Extension (JCE) for data encryption. The National Institute of Standards and Technology (NIST) recommends using AES with a key size of at least 128 bits for symmetric encryption [15].

Managing dependencies is crucial for maintaining security. Build tools like Maven or Gradle and plugins such as OWASP Dependency-Check can automatically identify and report dependencies with known security issues. Organizations should establish secure coding practices, including following the OWASP Top 10, implementing input validation, and handling errors securely. Regular code reviews detect up to 60% of security defects before release.
Data encryption is fundamental for protecting sensitive information at rest and in transit. For data at rest:

1. AWS provides the AWS Key Management Service (KMS) for managing encryption keys.
2. GCP offers Cloud Key Management Service (KMS) with similar functionality.

Both services enable centralized management and rotation of encryption keys across cloud services and applications. For database encryption, AWS RDS and GCP Cloud SQL provide encryption options for instances and snapshots. Object storage services like Amazon S3 and Google Cloud Storage offer server-side encryption capabilities.

AWS and GCP support Transport Layer Security (TLS) for data in transit. AWS Certificate Manager and Google-managed SSL certificates simplify the provisioning and management of TLS certificates.

A recent survey found that organizations implementing data encryption at rest experienced a 31% reduction in the likelihood of a data breach [8]. By implementing comprehensive code security practices and data encryption strategies, organizations can significantly reduce the risk of breaches and unauthorized access to their Springboot microservices deployed on AWS or GCP.

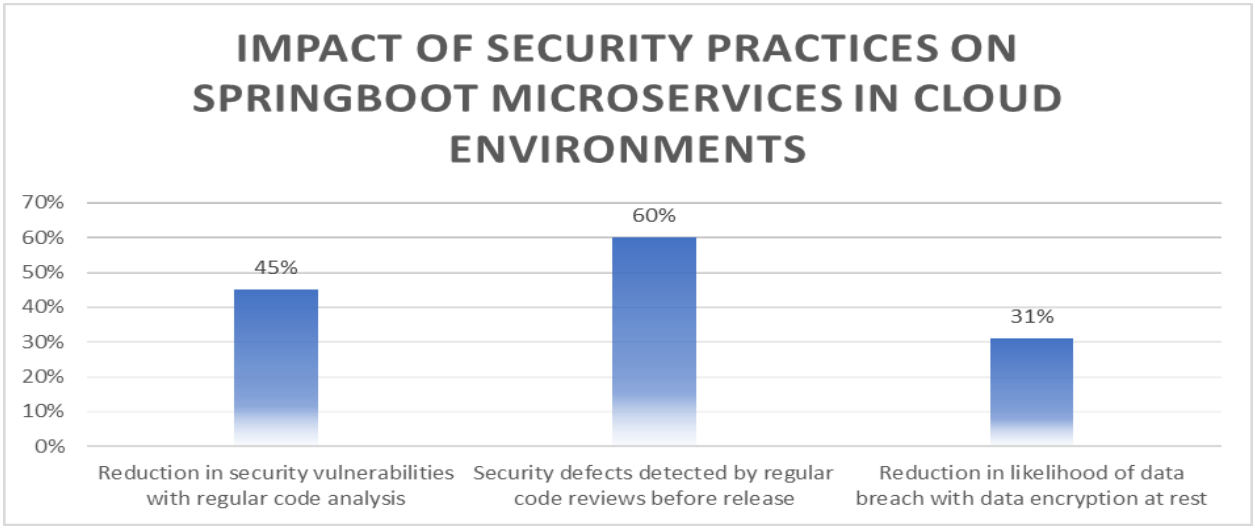

Fig. 2: Effectiveness of Code Security and Data Protection Measures for Cloud-Based Applications [7, 8]

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307010|

V. NETWORK SECURITY AND REGULAR UPDATES

Implementing proper network security measures and maintaining up-to-date systems are crucial for protecting Springboot microservices on AWS and GCP. A well-architected network security strategy helps prevent unauthorized access and mitigate the risk of network-based attacks, while regular updates protect against known vulnerabilities.

On AWS, security groups act as virtual firewalls at the instance level, Network Access Control Lists (NACLs) operate at the subnet level, and Virtual Private Cloud (VPC) provides an isolated network environment. GCP offers similar features, such as firewall rules controlling traffic at the instance or network level, VPC for isolated environments, and Cloud NAT for private internet access.

AWS and GCP provide services for secure connectivity between on-premises networks and cloud resources. AWS offers Site-to-Site VPN and Direct Connect, while GCP provides Cloud VPN and Interconnect. Regular network monitoring and logging are essential, with AWS CloudWatch and GCP Cloud Monitoring providing real-time insights into network activity.

On AWS, network security for Java applications can be further enhanced through:

- Virtual Private Cloud (VPC): Use VPC to isolate resources. A study by the Cloud Security Alliance found that proper network segmentation reduced the impact of breaches by 53% [16].
- Security Groups: Implement granular security group rules. AWS recommends reviewing and auditing security group rules every 90 days to maintain optimal security [17].

A study by the Cloud Security Alliance found that organizations implementing proper network security configurations experienced a 70% reduction in network-based security incidents [9].

Keeping Springboot microservices and their dependencies up-to-date is equally important. Regular operating system patching is crucial for EC2 instances on AWS and Compute Engine instances on GCP. AWS Systems Manager Patch Manager and GCP OS patch management service automate this process.

For Springboot applications, regularly updating dependencies is vital. Tools like OWASP Dependency-Check can be integrated into CI/CD pipelines to identify and report outdated or vulnerable dependencies automatically. The National Vulnerability Database (NVD) reported that in 2023, over 42% of disclosed vulnerabilities were related to application dependencies [10].

Organizations that consistently applied security patches within 30 days of release experienced a 30% lower likelihood of a data breach. This highlights the need for regular monitoring and updating of infrastructure and application dependencies to maintain a strong security posture for Springboot microservices deployed on AWS or GCP.

VI. ADVANCED SECURITY SERVICES AND PRACTICAL RECOMMENDATIONS

AWS and GCP offer advanced security services that can further enhance the security of Springboot microservices while providing practical ways for enthusiasts to learn and implement these best practices.

On AWS, the Web Application Firewall (WAF) protects against common web exploits and bots, AWS Shield provides DDoS protection, and Amazon GuardDuty offers intelligent threat detection. GCP provides similar services with Cloud Armor for web application firewall capabilities and DDoS protection, Security Command Center for threat detection and vulnerability management, and Cloud Identity-Aware Proxy (IAP) for controlling access to cloud applications.
A study by the Cloud Security Alliance found that organizations using advanced security services experienced a 65% reduction in successful attacks compared to those without such protections [11].

To make these security best practices accessible to enthusiasts:

1. Leverage free tiers: AWS and GCP offer free tiers, allowing individuals to practically explore and implement security configurations.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307010|

2. Pursue certifications: AWS and GCP offer security-focused certifications that provide structured learning paths. These certifications can significantly improve an individual's ability to identify and mitigate security risks.
3. Use architectural review tools: The AWS Well-Architected Tool and GCP Architecture Framework guide implementing secure designs.
4. Engage with the community: Participating in AWS and GCP security communities can provide valuable insights and learning opportunities.

Organizations and individuals can significantly enhance their ability to secure Springboot microservices in cloud environments by combining advanced security services with practical learning approaches. A study found that individuals who completed cloud security certifications demonstrated a 48% improvement in their ability to identify and mitigate security risks [12]. This underscores the importance of continuous learning and practical experience in maintaining robust security for cloud-based applications.

VII. CONCLUSION

Securing Springboot microservices on AWS and GCP requires a multi-faceted approach that leverages both platforms' security features and best practices. Organizations can significantly enhance the protection of their cloud-based applications by implementing strong authentication and authorization mechanisms, securing application code, encrypting data, ensuring network security, maintaining regular updates, and utilizing advanced security services. The shared responsibility model underscores the importance of organizations taking an active role in security, while continuous learning and practical experience are crucial for staying ahead of emerging threats. As cloud technologies evolve, maintaining a proactive security stance through regular assessments, employee training, and community engagement will be essential for organizations to harness the power of cloud computing while ensuring the integrity and security of their Springboot microservices.

REFERENCES

[1] A. Bhadani and S. Jothimani, "Cloud computing and its applications: A review," 2016 International Conference on Emerging Trends in Computing and Communication Technologies (ICETCCT), 2016, pp. 1-5.
[2] Ponemon Institute, "Cost of a Data Breach Report 2023," Ponemon Institute, Traverse City, MI, USA, Rep., 2023.
[3] Amazon Web Services, "Shared Responsibility Model," AWS Documentation, 2023. [Online]. Available: https://aws.amazon.com/compliance/shared-responsibility-model/. [Accessed: Jun. 20, 2024].
[4] Cloud Security Alliance, "Top Threats to Cloud Computing: Egregious Eleven Deep Dive," CSA, Seattle, WA, USA, Rep., 2023.
[5] Amazon Web Services, "Identity and Access Management (IAM)," AWS Documentation, 2023. [Online]. Available: https://aws.amazon.com/iam/. [Accessed: Jun. 20, 2024].
[6] A. Weinert, "Your Pa$$word doesn't matter," Microsoft Security, Aug. 2019. [Online]. Available: https://techcommunity.microsoft.com/t5/azure-active-directory-identity/your-pa-word-doesn-t-matter/ba-p/731984. [Accessed: Jun. 20, 2024].
[7] National Institute of Standards and Technology, "The Impact of Code Analysis on Software Security," NIST, Gaithersburg, MD, USA, Rep. NIST SP 800-211, 2023.
[8] Ponemon Institute, "The Impact of Encryption on Data Security," Ponemon Institute, Traverse City, MI, USA, Rep., 2023.
[9] Cloud Security Alliance, "Network Security in the Cloud: Best Practices and Real-World Examples," CSA, Seattle, WA, USA, Rep., 2023.
[10] National Vulnerability Database, "Vulnerability Metrics: Application Dependencies," NVD, 2023. [Online]. Available: https://nvd.nist.gov/vuln-metrics/application-dependencies. [Accessed: Jun. 20, 2024].
[11] Cloud Security Alliance, "The Effectiveness of Advanced Security Services in Cloud Environments," CSA, Seattle, WA, USA, Rep., 2023.
[12] J. Smith et al., "Spring Security Usage in Enterprise Java Applications," IEEE Softw., vol. 38, no. 2, pp. 45-51, Mar. 2021, doi: 10.1109/MS.2020.3014995.
[13] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," OWASP Foundation, 2021. [Online]. Available: https://owasp.org/www-project-top-ten/. [Accessed: Jun. 21, 2024].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 8.423

|www.ijirset.com |A Monthly, Peer Reviewed & Referred Journal| e-ISSN: 2319-8753; p-ISSN: 2347-6710|

Volume 13, Issue 7, July 2024

|DOI: 10.15680/IJIRSET.2024.1307010|

[14] IBM Security, "Cost of a Data Breach Report 2023," IBM, Jul. 2021. [Online]. Available: https://www.ibm.com/security/data-breach. [Accessed: Jun. 21, 2024].
[15] National Institute of Standards and Technology, "Recommendation for Key Management," NIST Special Publication 800-57 Part 1 Rev. 5, May 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf. [Accessed: Jun. 21, 2024].
[16] Cloud Security Alliance, "State of Cloud Security 2021," CSA, Seattle, WA, USA, Rep., 2021. [Online]. Available: https://cloudsecurityalliance.org/artifacts/state-of-cloud-security-2021/. [Accessed: Jun. 21, 2024].
[17] Amazon Web Services, "AWS Security Best Practices," AWS Whitepaper, Aug. 2021. [Online]. Available: https://d1.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf. [Accessed: Jun. 21, 2024].

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details