Rebuilding the Architecture with Security Best Practices

Purpose

Now that we've explored common misconfigurations in cloud deployments by intentionally building an insecure version of SecureCart, the next step is to redesign it with a secure architecture. This will demonstrate AWS security best practices in a real-world scenario.

What You'll Build

A custom VPC with both public and private subnets

EC2 instance (Node.js app) hosted in a private subnet

RDS (MySQL) database hosted in a private subnet

Application Load Balancer (ALB) in a public subnet

Internet Gateway and NAT Gateway for controlled internet access

AWS WAF integrated with ALB for application-layer protection

Key Security Learnings

Only ALB is exposed to the public internet

EC2 and RDS reside in private subnets

Principle of Least Access enforced using Security Groups

NAT Gateway provides controlled outbound access to EC2

AWS WAF filters HTTP traffic to block SQL injection, XSS, bots, and other threats

Why This Matters

Security is not just about using advanced tools - it's about designing cloud architecture that minimizes risk by default.

In this project, we implemented a layered security model:

Network Isolation: EC2 and RDS are placed in private subnets to eliminate direct internet exposure.

Controlled Access: Security Groups are configured to allow only the required traffic - e.g., only ALB can reach the app server, and only the app server can reach the database.

Defense-in-Depth: Even if one layer is compromised, others (like WAF or subnet restrictions) continue to protect the application.

AWS WAF: Adds an application-layer firewall that blocks known threats like:

SQL injection attacks

Cross-site scripting (XSS)

Malicious IPs and bot traffic

Access to sensitive endpoints (e.g., admin panels)

This design reflects industry best practices and gives you hands-on experience with securing cloud-native applications. It’s how real production systems are built and protected on AWS.