

Real Task:

How We Blocked Country-Level Traffic to a Global Cloud Run App

Mohamed Azarudeen M

5th July, 2025

Introduction

Why Block Users from a Particular Country?

- Compliance with Government Regulations
- Legal & Data Privacy Restrictions
- Prevent Security Threats from High-Risk Regions
- Business Decisions
- Protect Against DDoS and Spam Attacks

Example Scenarios:

Scenario	Country Blocking Example
<i>US Export Restrictions</i>	Block North Korea, Iran, Syria
<i>GDPR Non-compliance</i>	Block all EU countries
<i>Focus on local/regional markets only</i>	A startup in India blocks other regions
<i>Continuous bot attacks from specific regions</i>	Block Russia, China for admin panels
<i>Online banking app targeting only one country</i>	Allow only India, block all others



What is Google Cloud Armor?

Google Cloud Armor is a **security service** from Google Cloud Platform (GCP) that protects your applications from unwanted traffic at the network level, even before it reaches your application.

Where Does Cloud Armor Fit in the Architecture?

In cloud applications, we often use a **Load Balancer** to expose the app to the public internet.

Cloud Armor acts as a **firewall in front of the Load Balancer** and **filters incoming traffic** based on:

- Geographic location (Geo Blocking)
- IP address ranges
- Security policies
- Known vulnerability patterns (Layer 7 WAF)

Why Do We Use Cloud Armor?



- ✓ Protect against malicious traffic like bots, DDoS attacks.
- ✓ Restrict traffic from specific countries (Geo-blocking).
- ✓ Ensure only allowed users or networks access your apps.
- ✓ Improve security without changing application code.

Problem Statement

We deployed a **microservice on Google Cloud Run**, exposed through a **Global HTTPS Load Balancer** making it publicly available worldwide.

Our requirement was to **restrict users from China** from accessing this global endpoint, due to business restrictions.

Key constraint:

-  We couldn't change the app code or deploy new middleware.
-  Solution had to be applied **at the infrastructure layer**.

Solution Architecture

Component	Purpose
Google Cloud Run	Hosts the microservice (serverless)
HTTPS Global Load Balancer	Exposes the Cloud Run service to the internet
Cloud Armor	Blocks China traffic using geo-based policies
Terraform	Manages the entire setup as Infrastructure as Code
Cloud Logging	Monitors blocked and allowed traffic

Implementation Step

Step 1: Create a Cloud Armor Security Policy

This policy will block requests from **country code 'CN' (China)**.

```
resource "google_compute_security_policy" "block_china" {
  name          = "block-china-traffic"
  description    = "Block China"
  rule {
    priority = 1000
    description = "Deny users from China"
    match {
      versioned_expr = "SRC_IPS_V1"
      expr {
        expression = "request.geo.country == 'CN'"
      }
    }
    action = "deny(403)"
    preview = true # Start in preview to monitor before blocking
  }
  rule {
    priority = 2147483647
    description = "Allow everyone else"
    match {
      versioned_expr = "SRC_IPS_V1"
      config {
        src_ip_ranges = ["*"]
      }
    }
    action = "allow"
  }
}
```

✓ Step 2: Attach Policy to the Load Balancer's Backend Service

The Load Balancer connects to Cloud Run via a **serverless NEG backend service**.

Attach the security policy like this:

```
resource "google_compute_backend_service" "cloudrun_backend" {
  name          = "cloudrun-backend"
  security_policy = google_compute_security_policy.block_china.id
  # existing configurations
}
```

✓ Step 3: Apply Terraform & Validate

Deploy the changes:

```
terraform plan
terraform apply
```

Monitor requests using **Cloud Logging**. Example query to find China-based access:

```
resource.type="http_load_balancer"
jsonPayload.enforcedSecurityPolicy.name="block-china-traffic"
jsonPayload.geoLocation.country="CN"
```

✓ Step 4: Enforce the Block

Once verified, remove `preview = true`:

```
preview = false
```

Re-apply Terraform to **block China users effectively**.

Rollback Plan

Scenario	Rollback Action
Too much traffic blocked accidentally	Remove/adjust the blocking rule in Terraform
Temporary unblock needed	Set <code>preview = true</code> to disable blocking
Remove country restrictions entirely	Detach the security policy from the backend

Final Results

How Would You Do This in AWS?

In AWS, you'd implement the **same requirement** using these services:

GCP Component	AWS Equivalent
Cloud Armor	AWS WAF
Global HTTPS Load Balancer	Amazon CloudFront + Application Load Balancer
Terraform	Terraform (AWS Provider) or CloudFormation
Cloud Logging	Amazon CloudWatch Logs
Cloud Run (serverless app)	AWS Lambda + API Gateway, or ECS Fargate



AWS Implementation Outline:

1. Create an **AWS WAF Web ACL**.
2. Add a **geo match condition for country = CN**.
3. Attach the WAF to an **ALB** or **CloudFront distribution**.
4. Deploy using Terraform or AWS Console.

Final Takeaways

- ✓ Cloud Armor provides a powerful **network-level security** without app changes.
- ✓ Geo-blocking is essential for compliance in global apps.
- ✓ Start in preview mode to monitor impact.
- ✓ Same concept applies in **AWS, Azure, and other cloud providers**.

