
🛡️ NIST AI RMF Implementation Playbook

Project: Secure AI Gateway Implementation

Framework: NIST AI Risk Management Framework (Functions: Manage, Measure)

Infrastructure: Cloudflare AI Gateway, Workers AI (Llama-3), Python

---

Phase 1: The Objective

To move from "Direct API Access" (High Risk, No Visibility) to a **Managed AI Pipeline** (Low Risk, High Visibility).

- **Goal:** Intercept, log, and control all traffic sent to Large Language Models (LLMs).
- **NIST Alignment:**
  - **Manage:** Control access via Gateway authentication.
  - **Measure:** Log prompts, tokens, and latency for auditability.

Phase 2: Account Discovery (The "ID Trap")

Before writing code, we had to locate the correct infrastructure identifiers. *Crucial Lesson: Do not confuse Zone IDs with Account IDs.*

Step 1: Locate Account ID

1. Log in to Cloudflare Dashboard.
2. Look at the browser URL: https://dash.cloudflare.com/YOUR_ACCOUNT_ID/ai/gateway
3. **Artifact:** The 32-character string after .com/ is the **Account ID**.
   - *Correct:* c267... (From URL)
   - *Incorrect:* Any ID found on the "Website Overview" sidebar (Zone ID).

Phase 3: Authentication Strategy

We attempted "Least Privilege" (Tokens) but fell back to "Administrator Access" (Global Key) due to scope friction.

Method A: API Tokens (Preferred for Prod)

- **Path:** User Profile > API Tokens > Create Token.
- **Permissions Required:**
  - Account > AI Gateway > Run
  - Account > Workers AI > Read
- **The Critical Fix:** You must select **"All Accounts"** under *Account Resources* to prevent scoping errors (Error 10000).

Method B: Global API Key (The "Master Key")

- **Used For:** Debugging when Tokens fail with Authentication Error (10000).
- **Path:** User Profile > API Tokens > API Keys > Global API Key.
- **Headers Used:** X-Auth-Email and X-Auth-Key.

Phase 4: Gateway Infrastructure Setup

We established the control plane.

Step 1: Create the Gateway

1. Navigate to **AI** > **AI Gateway**.
2. Click **Create Gateway**.
3. **Name:** secure-scanner-gateway.
4. **Action:** This generates a "Universal Endpoint" URL: https://gateway.ai.cloudflare.com/v1/{ACCOUNT_ID}/{GATEWAY_NAME}

Step 2: The "Unlock" (Crucial Fix)

- **Issue:** The Gateway was created with "Authentication Enabled" (Lock Icon 🔒).
- **Symptom:** All API calls returned 403 Forbidden or Unauthorized.
- **The Fix:**
  1. Click secure-scanner-gateway.
  2. Go to **Settings**.
  3. **Toggle OFF** "API Authentication" (Enforce Access Control).
  4. **Verify:** The Lock Icon 🔒 must disappear from the dashboard list.

Phase 5: The Implementation (Python)

We built a script to route prompts through the Gateway instead of hitting the model directly.

The Final Artifact (ai_ops_test.py)

This is the working code block used to validate the pipeline.

Python

```
import requests

# --- CONFIGURATION ---
GATEWAY_NAME = "secure-scanner-gateway"
ACCOUNT_ID = "c267966add7c04e16295a38bef315e66"
EMAIL = "tharunrs007@gmail.com"
GLOBAL_KEY = "PASTE_YOUR_GLOBAL_KEY_HERE"  # <--- The Master Key
# ---------------------

def run_pipeline():
    print(f"🚀 Targeting Gateway: {GATEWAY_NAME}")

    # The Gateway URL (Not the Dashboard URL!)
    url = f"https://gateway.ai.cloudflare.com/v1/{ACCOUNT_ID}/{GATEWAY_NAME}/workers-ai/@cf/meta/llama-3-8b-instruct"

    headers = {
        "X-Auth-Email": EMAIL,
        "X-Auth-Key": GLOBAL_KEY.strip(),
        "Content-Type": "application/json"
    }

    payload = {
        "messages": [
            {"role": "system", "content": "You are a specific security bot."},
            {"role": "user", "content": "Confirm system operational."}
        ]
    }

    try:
```

```
        response = requests.post(url, headers=headers, json=payload)
        if response.status_code == 200:
            print("✅ VICTORY! Response received through Gateway.")
            print(f"🤖 AI: {response.json()['result']['response']}")
        else:
            print(f"❌ Error {response.status_code}: {response.text}")
    except Exception as e:
        print(f"❌ Script Failed: {e}")

if __name__ == "__main__":
    run_pipeline()
```

Phase 6: Verification (NIST "Measure")

We confirmed the system works by checking the telemetry.

1. **Action:** Run the Python script.
2. **Verify:** Go to Cloudflare Dashboard > AI Gateway > secure-scanner-gateway > **Logs**.
3. **Evidence:**
    - **Prompt:** Visible ("Confirm system operational").
    - **Status:** Success (200 OK).
    - **Metadata:** Token count and Latency are recorded.

Appendix: Troubleshooting Log

This section records the errors we faced so we recognize them next time.

Error Code	Meaning	Root Cause	The Fix
10000	Authentication Error	Token Scope Mismatch	Used "Global API Key" or recreated token with "All Accounts" scope.
401	Unauthorized	Wrong Credentials	Verified Account ID vs. Zone ID; switched to Global Key.

2001	Gateway Not Configured	Script tried to use a non-existent gateway	Created Gateway manually in Dashboard to accept TOS.
403	Forbidden	Gateway Locked 🔒	Turned **OFF** "API Authentication" in Gateway Settings.
400	Bad Request	URL Error	Switched from dash.cloudflare.com URL to gateway.ai.cloudflare.com.

Phase 7: Availability Engineering (DoS Defense)

- **Objective:** Prevent resource exhaustion attacks by limiting request volume.
- **Configuration:**
  - **Mechanism:** Cloudflare Rate Limiting (Fixed Window).
  - **Policy:** 3 requests / 1 minute (Test Mode).
  - **Outcome:** Requests >3 receive `HTTP 429`.
- **Validation:**
  - Developed `dos_attack_test.py` to simulate high-frequency traffic.
  - Validated that the Gateway fails open (200 OK) for legitimate traffic and fails closed (429 Blocked) for abuse.

Phase 8: Confidentiality Engineering (Data Privacy)

- **Objective:** Prevent data leakage (PII, Credentials, Secrets) into persistent storage logs, ensuring compliance with privacy standards (GDPR/CCPA).
- **Configuration:**
  - **Mechanism:** Cloudflare Log Redaction.
  - **Action:** Disabled *"Store request and response payloads"* in Gateway Settings.
  - **Outcome:** Gateway processes prompts for metrics (tokens/cost) but **discards** the actual text content immediately.
- **Validation:**
  - Developed `privacy_test.py` to inject a "Canary Token" (fake secret password).
  - **Verified:** The Gateway processed the request (200 OK), but the Logs showed `[Redacted]` or empty fields for the message content.

Phase 9: Final Architecture Summary

- **Current State:**
  - **Access:** Locked via API Tokens/Global Keys (**Integrity**).
  - **Traffic:** Routed through `secure-scanner-gateway` (NIST **Manage**).
  - **Resilience:** Rate Limited to 3 req/min to block DoS attacks (**Availability**).
  - **Privacy:** Payload logging disabled to protect secrets (**Confidentiality**).
  - **Observability:** Full visibility into Token Usage, Latency, and Error Rates (NIST **Measure**).