|  |  |  |
| --- | --- | --- |
| Feature / Aspect | Option 1: Akamai G2O Header Auth with IP Restriction (Shape's IPs) | Option 2: Akamai G2O Header Authentication (No IP Restrictions) |
| Description | Traffic from Akamai hits your origin. Your origin only accepts requests if: 1. They originate from Shape Security's specific IP addresses (whitelisted). 2. They possess a valid Akamai G2O header. | Traffic from Akamai hits your origin. Your origin only accepts requests if: 1. They possess a valid Akamai G2O header. (No network-level IP filtering on the origin). |
| Pros | 1. Robust Layered Origin Access: Provides two distinct checks (IP and G2O signature) for traffic reaching your origin, making direct origin bypass extremely difficult for non-Akamai/Shape traffic. 2. Clear Firewall Enforcement: Easy to understand and implement a network-level "hard block" at the firewall/load balancer level for unauthorized IPs. | 1. Zero IP Management Overhead: Eliminates the significant operational burden of maintaining and updating dynamic Akamai/Shape IP address lists. 2. Reduced Risk of Self-Inflicted Outages: Prevents legitimate traffic blocks due to outdated IP whitelists. 3. Focus on Cryptographic Trust: Relies on the strong cryptographic guarantee of G2O, which is more reliable than static IP lists for a global CDN. 4. Scalability & Agility: Easier to scale Akamai/Shape deployment or changes without needing origin firewall modifications. |
| Cons | 1. High Maintenance Overhead: Akamai's and Shape's global IP ranges are vast and dynamic. Continuously managing and updating these IPs is a significant, error-prone, and ongoing operational burden. 2. High Risk of Service Outages: Failure to promptly update IP whitelists will block legitimate Akamai/Shape traffic, leading to a severe denial of service for your users. This is the most critical operational risk. 3. Doesn't Solve Token Compromise Issue Directly: While it prevents direct origin access, if the compromised token is used in a request that comes through Akamai and Shape (via their whitelisted IPs), this option offers no additional protection against the abuse of the token itself. 4. False Sense of Security Against Advanced Threats: The IP restriction offers no defense against sophisticated bots or human-driven attacks that successfully proxy through Akamai/Shape's legitimate infrastructure. | 1. Sole Reliance on G2O for Origin Gatekeeping: If G2O is misconfigured or a severe vulnerability allows its bypass, there's no immediate network-level "kill switch" for non-Akamai traffic trying to hit the origin directly. However, G2O is designed to be robust. 2. Doesn't Solve Token Compromise Issue Directly: Similar to Option 1, G2O authenticates the source (Akamai), but doesn't, by itself, validate the legitimacy of the token use within the application context. This is where Gigya's token validation and Shape's behavioral analysis come in. |
| Relevant Threats & Attacks (with Impact vs. Likelihood) |  |  |
| 1. API Token Compromise & Abuse (Scenario you experienced) | \* Impact: High (data modification, account takeover, fraud, reputational damage) \* Likelihood: Medium-High (if token validation is weak, or if XSS/credential stuffing leads to token theft) Mitigation: IP whitelisting does NOT directly mitigate this if the malicious request comes through Akamai/Shape's legitimate IPs. The attack is at the application layer, not the network layer in this specific context. Shape Security's analysis is the key mitigation here. | \* Impact: High \* Likelihood: Medium-High Mitigation: G2O does NOT directly mitigate this. The attack is at the application layer. This requires strong token validation from Gigya, and behavioral analysis from Shape to detect anomalous token usage or bot-driven attempts to obtain/use tokens. |
| 2. Direct-to-Origin Attack (Bypass CDN) | \* Impact: Low (unlikely to succeed) \* Likelihood: Low Mitigation: Excellent. Both IP whitelisting and G2O header validation must be bypassed, which is extremely difficult. | \* Impact: Medium-High (if successful, could expose origin directly) \* Likelihood: Low Mitigation: Good. G2O provides strong cryptographic protection. Bypassing G2O would require significant cryptographic expertise or a severe vulnerability/misconfiguration. |
| 3. Credential Stuffing/Account Takeover (ATO) | \* Impact: High (financial loss, data breach, reputational damage) \* Likelihood: High Mitigation: IP whitelisting offers NO direct mitigation against this, as these attacks often originate from distributed botnets that tunnel through legitimate CDN paths. Shape Security is the primary defense here. | \* Impact: High \* Likelihood: High Mitigation: G2O offers NO direct mitigation against this. Shape Security is your primary defense. Its behavioral analysis detects and blocks automated credential stuffing and ATO attempts by identifying non-human patterns, even if the traffic comes from "clean" IPs. |
| 4. Web Scraping/Data Exfiltration | \* Impact: Medium-High (competitive disadvantage, intellectual property loss, privacy violation) \* Likelihood: High Mitigation: IP whitelisting offers NO direct mitigation if the scraping traffic routes through Akamai/Shape. Shape Security is crucial here. | \* Impact: Medium-High \* Likelihood: High Mitigation: G2O offers NO direct mitigation. Shape Security is your primary defense, as it's designed to detect and block automated scraping activities. |
| 5. Accidental Self-Inflicted DoS (from IP list errors) | \* Impact: High (site outage, revenue loss, reputational damage) \* Likelihood: Medium-High (given the dynamic nature of CDN IPs and human error) Mitigation: High Risk. Requires constant, diligent IP list management. | \* Impact: Low \* Likelihood: Very Low Mitigation: Almost entirely eliminated. No IP list to manage. |
| Most Practical Option | Less Practical due to the severe operational burden, the high risk of outages, and its inability to directly address the sophisticated bot/fraud attacks (like the token abuse you experienced) that bypass simple IP checks. | Most Practical. It focuses on reliable cryptographic authentication (G2O) and allows Shape Security to perform its specialized, higher-value bot/fraud prevention without brittle network-level constraints. This frees up resources to focus on the application-layer security that was demonstrably exploited. |

### **Security Architect's Recommendation**

Given the recent incident involving authentication token abuse, the context points strongly towards **application-layer attacks and sophisticated automated fraud**, rather than simple network-level bypass attempts.

Therefore, **Option 2: Akamai G2O Header Authentication (No IP Restrictions)** is the **most practical and recommended solution** for your environment.

**Why Option 2 is superior in your context:**

1. **Addresses the Root Problem (via Shape, not IP):** The attack you experienced (token abuse to change customer details) is an application-layer issue. IP whitelisting is a coarse network control that does *not* prevent an attacker who has a valid token and is routing through Akamai/Shape's legitimate IPs. Shape Security is your key defense here, as it analyzes behavioral anomalies and bot patterns, which are essential for detecting and blocking such sophisticated attacks *even if they appear to originate from legitimate Akamai IPs*. By removing IP restrictions, you allow Shape to operate optimally without introducing unnecessary friction.
2. **Operational Efficiency and Resilience:** The maintenance overhead of managing Akamai's and Shape's dynamic IP ranges is substantial and a constant source of risk for outages. Eliminating this reduces the chances of self-inflicted wounds, allowing your security team to focus on proactive threat hunting and application-level security, which is where your recent breach occurred.
3. **Trust in Strong Cryptography:** G2O provides a robust cryptographic guarantee that traffic arriving at your origin from Akamai has not been tampered with and truly originated from Akamai. This is a far more reliable trust mechanism than a mutable list of IP addresses. If your origin successfully validates the G2O header, you can confidently assume the request came through Akamai.

**Key Mitigations & Next Steps (beyond the chosen option):**

Even with Option 2, the core issue of the "authentication token being used to make API calls and change customer details and credentials" needs direct attention from Gigya's implementation and your application logic.

1. **Deep Dive into Gigya Token Management:**

* **Token Validity & Expiration:** Are your access tokens short-lived? Are refresh tokens securely managed and rotated?
* **Token Revocation:** Can compromised tokens be immediately revoked by Gigya? Is your application effectively checking token revocation status on API calls?
* **Scope & Permissions:** Are tokens issued with the principle of least privilege? Does an API token have more permissions than strictly necessary to perform its intended function? For example, should a "read-only" token be able to change customer details?
* **Token Storage:** How are tokens stored on the client-side? Are they vulnerable to XSS (Cross-Site Scripting) or other client-side attacks?
* **Device Binding/Contextual Authentication:** Can you bind tokens to specific devices or IP addresses? If a token used for an API call comes from a wildly different IP or device fingerprint than where it was originally issued, can Gigya or your application flag it?

1. **Leverage Shape Security to its Full Potential:**

* **API Protection:** Ensure Shape Security is configured to protect *all relevant API endpoints*, not just web pages. Shape excels at identifying automated API abuse.
* **Behavioral Anomaly Detection:** Configure Shape to specifically look for anomalous behavior around API calls, especially those related to account modifications or credential changes. This includes:
* Unusual request rates to sensitive APIs.
* Repeated attempts to modify multiple user accounts from a single source.
* Changes in user-agent, device fingerprint, or other request characteristics for a given session.
* Rapid sequential API calls that mimic automation rather than human interaction.
* **Integration with SIEM/SOAR:** Ensure Shape's logs and alerts are integrated into your Security Information and Event Management (SIEM) and Security Orchestration, Automation, and Response (SOAR) platforms for real-time monitoring and automated response.

1. **Application-Layer Protections:**

* **Rate Limiting:** Implement strict rate limiting on all sensitive API endpoints (e.g., password change, email change, profile update) to mitigate brute-force attempts or rapid-fire token abuse.
* **Input Validation:** Ensure robust input validation for all API parameters to prevent injection attacks or malformed requests.
* **Logging and Auditing:** Enhance logging for all critical API calls, especially those modifying customer data or credentials. Log who made the call (user ID), originating IP (from Akamai's True-Client-IP header), timestamp, and parameters.
* **Multi-Factor Authentication (MFA) for Sensitive Actions:** For critical actions (e.g., changing email, password, adding payment methods), consider requiring a second factor of authentication (even if the user is already logged in with a valid token). Gigya supports this.

By adopting Option 2, you streamline your network-level security and free up resources to properly configure and leverage your advanced security tools (Gigya and Shape) to address the sophisticated application-layer attacks that are your primary concern.