Technical Documentation – Voice Agent Application

Overview

We have developed a Voice Agent Application designed to streamline the process of creating, managing, and deploying AI-powered voice assistants. The application integrates several third-party services to provide robust telephony and communication functionalities, while leveraging modern AI development tools for rapid delivery.

Core Features

User Authentication

Login/Sign up with secure email verification.

Postmark is used for email authentication and transactional communications.

Voice Agents

Users can create and configure AI-powered agents capable of making calls, handling conversations, and responding dynamically.

Telephony Capabilities

Number Purchasing: Users can buy virtual phone numbers through Twilio.

Outbound Calls: Agents can place calls to external recipients.

Inbound Calls: Agents can receive and handle incoming calls.

Call Forwarding: Calls can be forwarded to other numbers for seamless connectivity.

AI Voice Integration

11labs is used to provide high-quality, natural-sounding AI-generated voices for the agents.

Voice Assistant Orchestration

Vapi is integrated to manage conversation flows, intent handling, and real-time agent performance.

Development Environment

Platform: Replit (AI-powered coding agent).

Reason for Choice: Replit enabled fast prototyping and development under a tight deadline.

Capabilities:

Takes natural language prompts and generates code.

Manages project file structures automatically.

Provides an execution and deployment environment.

API Keys & Security Management

Security of credentials (API keys, external connection URLs, and sensitive environment variables) was a priority during development.

Key Management Practices:

Secrets Storage:

All API keys (11labs, Vapi, Twilio, Postmark, etc.) were stored in Replit's Secret Environment Variables feature.

This prevents credentials from being exposed into source code.

Server-Side Usage Only:

Keys were only used in server-side scripts, never in client-side code.

This ensures keys cannot be exposed through browser access or frontend inspection.

Private Repository Management:

The application was maintained in private Replit projects to prevent unauthorized access.

Public clones were explicitly avoided, as public projects on Replit expose code and allow others to view and modify it.

Deployment and Transfer to Client

The app was initially deployed in our Replit account more than three months ago without any security incidents.

When transferring the app to the client's Replit:

The client's account was not upgraded, which meant private cloning was initially not possible.

We requested an account upgrade so that the app could be transferred securely.

After the upgrade, a private clone was created, ensuring only the client had access.

On first launch in the client's Replit, the environment prompted for API keys.

Keys were securely stored in the client's secret variables to ensure proper functioning.

Reported Incident – Client's Claim of Key Compromise

After the transfer, the client reported that API keys had been compromised. Below is our assessment of the case:

Timeline:

The app ran securely in our account for 3+ months without issue.

The compromise claim only arose after cloning the app into the client's Replit account.

Security Measures Taken on Our Side:

The app was always private.

Transferred via private cloning, not public clone.

API keys were stored exclusively in Replit's secrets manager.

Keys were used only in server-side scripts.

Uncertainty of Compromise Source:

Since the cloned app in the client's account uses the same code and structure, the security posture should remain identical.

We cannot determine how the keys were exposed, as browser-side exposure is not possible given the server-side integration.

From our end, all industry-standard precautions were taken.

Conclusion

The Voice Agent App was developed with a strong emphasis on security, using Replit secrets management, private repositories, and server-side-only credential usage.

While the client reports compromised API keys after the transfer, we confirm that:

The application remained secure during our deployment.

Transfer was done securely using private cloning.

Keys were properly stored in secret variables.

Therefore, based on our practices and transfer process, the source of compromise cannot be attributed to our development or transfer method.