کنترل‌های امنیتی API در سیستم CI/CD

مقدمه

امنیت APIها در یک سیستم CI/CD، به ویژه در مونوریپو `aladdin-sandbox` که شامل چندین سرویس و تعامل با پلتفرم‌های خارجی است، از اهمیت بالایی برخوردار است. هرگونه ضعف امنیتی در این بخش می‌تواند منجر به دسترسی غیرمجاز، نشت اطلاعات حساس یا اختلال در سرویس‌ها شود. سیستم CI/CD پیاده‌سازی شده، با تمرکز بر اصول امنیت، چندین لایه کنترل را برای محافظت از APIها و تعاملات مرتبط با آن‌ها اعمال می‌کند.

1. مدیریت ایمن Secrets (اطلاعات حساس)

مهمترین جنبه امنیت API، نحوه مدیریت اطلاعات حساس مانند کلیدهای API، توکن‌ها و رمزهای عبور است. سیستم ما از روش‌های زیر برای محافظت از این اطلاعات استفاده می‌کند:

- **GitHub Secrets**: تمامی کلیدهای API (مانند `RENDER_API_KEY` که برای تعامل با Render.com) ذخیره می‌شوند. این روش GitHub Secrets و سایر اطلاعات حساس به عنوان (استفاده می‌شود مزایای زیر را دارد:
  - **عدم قرارگیری در کدبیس**: Secrets هرگز به صورت مستقیم در کد، فایل‌های پیکربندی یا تاریخچه Git ذخیره نمی‌شوند.
  - **دسترسی محدود**: این Secrets فقط در زمان اجرای GitHub Actions Workflow و توسط Jobهای مجاز قابل دسترسی هستند.
  - **رمزنگاری**: GitHub این Secrets را به صورت رمزنگاری شده ذخیره می‌کند و فقط در زمان اجرا به صورت متغیر محیطی در دسترس Job قرار می‌دهد.
  - **متغیرهای محیطی (Environment Variables)**: در زمان اجرای Secrets، Workflow به عنوان متغیرهای محیطی به `mamos_runner.py` یا سایر اسکریپت‌ها تزریق می‌شوند. این امر تضمین می‌کند که اطلاعات حساس فقط در حافظه و برای مدت زمان اجرای Job وجود دارند و پس از اتمام Job از بین می‌روند.

2. کنترل دسترسی و مجوزها (Access Control and Permissions)

سیستم CI/CD ما از مکانیزم‌های کنترل دسترسی دقیق برای محدود کردن اختیارات Workflowها و Jobها استفاده می‌کند:

- **`permissions` در GitHub Actions**: هر Workflow و Job در GitHub Actions دارای یک بلاک `permissions` مانند) است که به صراحت مشخص می‌کند چه مجوزهایی برای دسترسی به منابع مخزن (`contents: write`، `pull-requests: write`، `checks: write`) نیاز دارد. این رویکرد **اصل حداقل امتیاز (Principle of Least Privilege)** فقط به حداقل Job را پیاده‌سازی می‌کند، به این معنی که هر مجوزهای لازم برای انجام وظیفه خود دسترسی دارد.

- **توکن GitHub** : `GITHUB_TOKEN` به طور خودکار یک `GITHUB_TOKEN` موقت برای هر Job ایجاد می‌کند. این توکن دارای مجوزهای محدودی است که توسط بلاک `permissions` کنترل می‌شود و پس از اتمام Job منقضی می‌شود. این امر از سوءاستفاده احتمالی از توکن‌ها جلوگیری می‌کند.

3. امنیت تعامل با APIهای خارجی

برای انجام عملیات (مانند Render.com API) های خارجیAPI مسئول تعامل با `mamos_runner.py` استقرار است. امنیت در این تعاملات به شرح زیر تضمین می‌شود:

- **استفاده از HTTPS**: تمامی ارتباطات با APIهای خارجی از طریق پروتکل امن HTTPS انجام می‌شود. این امر تضمین می‌کند که داده‌ها در حین انتقال رمزنگاری شده و از حملات شنود (eavesdropping) محافظت می‌شوند.
- **اعتبارسنجی API Key**: `mamos_runner.py` از `RENDER_API_KEY` برای اعتبارسنجی درخواست‌ها به Render.com API استفاده می‌کند. این کلید به عنوان یک Secret مدیریت شده و فقط در زمان اجرای Job در دسترس است.
- **مدیریت خطا و گزارش‌دهی**: در صورت بروز خطاهای امنیتی (مانند `Client Error: Unauthorized 401` که در تست‌ها مشاهده شد)، `mamos_runner.py` خطا را شناسایی کرده و گزارش می‌دهد. این امر به تیم امکان می‌دهد تا به سرعت مشکلات امنیتی مربوط به اعتبارسنجی API را تشخیص داده و رفع کنند.

4. ایزوله‌سازی محیطی

تفکیک محیط‌ها به جلوگیری از نشت اطلاعات حساس و تداخل بین پیکربندی‌های امنیتی کمک می‌کند:

- **محیط‌های GitHub Actions (Environments)**: با تعریف محیط‌های `Staging` , `Test` و `Production` در GitHub Actions، می‌توان متغیرهای محیطی و Secrets خاص هر محیط را تعریف کرد. این بدان معناست که کلیدهای API پروداکشن هرگز در محیط استیجینگ یا تست در دسترس نیستند و بالعکس.
- **Jobهای ایزوله**: هر GitHub Actions در یک محیط ایزوله اجرا می‌شود. این ایزوله‌سازی Job نمی‌تواند به متغیرهای محیطی یا فایل‌های Job دیگر دسترسی پیدا کند، حتی اگر در یک Workflow باشند.

5. کنترل‌های انسانی و بازبینی

علاوه بر کنترل‌های خودکار، لایه‌های انسانی نیز برای افزایش امنیت APIها وجود دارد:

- **تایید دستی (Manual Approval)**: برای استقرار در محیط‌های حساس مانند `Staging` و `Production` ، نیاز به تایید دستی وجود دارد. این مرحله به تیم‌های امنیتی یا مسئولان محصول فرصت می‌دهد تا قبل از انتشار نهایی، تمامی جنبه‌های امنیتی، از جمله پیکربندی‌های API و دسترسی‌ها، را بازبینی کنند.
- **بازبینی کد (Code Review)**: هرگونه تغییر در کد که بر نحوه تعامل با APIها یا مدیریت Secrets تأثیر می‌گذارد، باید توسط همکاران بازبینی شود. این بازبینی‌ها به شناسایی آسیب‌پذیری‌های احتمالی قبل از ادغام کد در شاخه اصلی کمک می‌کند.

6. امنیت در سطح کد (Code-Level Security)

اگرچه سیستم CI/CD به طور مستقیم کد API را نمی‌نویسد، اما زیرساخت لازم برای اعمال بهترین شیوه‌های امنیتی در کد را فراهم می‌کند:

- **تحلیل استاتیک کد (Static Code Analysis)**: می‌توان ابزارهای تحلیل استاتیک کد را به خط لوله CI/CD اضافه کرد تا آسیب‌پذیری‌های امنیتی رایج (مانند تزریق SQL، XSS) در کد API شناسایی شوند.
- **اسکن وابستگی‌ها (Dependency Scanning)**: ابزارهایی برای اسکن وابستگی‌های پروژه‌ها (مانند کتابخانه‌های پایتون یا پکیج‌های Node.js) برای شناسایی آسیب‌پذیری‌های شناخته شده می‌توانند در CI/CD ادغام شوند.

نتیجه‌گیری

سیستم CI/CD پیاده‌سازی شده برای `aladdin-sandbox` با استفاده از ترکیبی از مدیریت ایمن Secrets، کنترل دسترسی دقیق، ایزوله‌سازی محیطی، کنترل‌های انسانی و امکان ادغام ابزارهای امنیتی در سطح کد، یک رویکرد چندلایه و قوی برای تضمین امنیت APIها ارائه می‌دهد. این کنترل‌ها به کاهش ریسک‌های امنیتی کمک کرده و اطمینان می‌دهند که APIها به صورت ایمن و قابل اعتماد عمل می‌کنند.