Safe Family - Intel (China) Co., Ltd.

2016-03 to 2017-03

Description: A cross-platform home device protection system that supports Android, iOS, and Windows platforms.

- Features include: application availability control; web page URL access control (Android and Windows platforms only); device time control (Android and iOS platforms only); electronic map fence settings (Android and iOS platforms only); New application, access to web page URL, instant notification of device time and instant response control; timely reporting of device address changes and timely notification of device access to electronic map fence; activity history and review; device location tracking in time.
- The account system is an account for each family. The account is divided into two groups of parents and children. Each family member has a limited number of family members. Each parent and child can have multiple devices, each parent. The device can be used to control all the devices of the child at home. Each child's device on the same platform shares the same application availability control, web page URL access control, and electronic map fence access notification. Each child shares the same device. The device can be controlled using time of day.
- The Android app supports configuration availability control based on application classification, iOS app supports configuration of availability control based on age, all platform applications support separate configuration availability control, and instant authorization for short periods of time or long-term application availability. The device can be configured for hours of workdays and non-working days, and you can configure which time periods you can use the device. The electronic map fence is set in the open map to select the fence center and square radius configuration parameters, and can be configured with an unlimited number of electronic map fences.
- For more details of the project, please refer to the official website of the project and the client description of each platform (see the link below).

Working as back-end developer/leader (back-end named CloudServices).

- The entire backend is divided into multiple different services running on different machines (or AWS EC2 instances), with different services behind reverse proxy/load balancing (local development and test environments using haproxy as reverse proxy and service) Request forwarding routing as an alternative), each service (in addition to the task scheduling Scheduler service) runs multiple Python/Tornado server examples and places nginx in front of it as load balancing (due to Python multithreading/multiprocess issues). Python/Tornado process instances for each background service are managed through Supervisor.
- Background services include: Auth service for authentication and authorization, MDM service for synchronizing iOS device control rule change commands, Family service for each client interface operation, for synchronizing various control rules and collecting

customers The Context service of the end data, the Scheduler service for periodically initiating the calculation of the change rule of the iOS device control rule, the ActionTrigger service for calculating the change of the iOS device control rule, and the mail template service for generating the mail content with the internationalized multi-language support. The Auth service and the MDM service are generally deployed in pairs on the same machine (or AWS EC2 instance), and the mail template service for generating mail content with internationalized multi-language support is provided by the official website (part-time).

- The background and the platform clients mainly use the HTTP protocol for communication, use the JSON format as the data exchange format, and use JSON Schema to verify the legality of the transmitted data. The message notification integration of each platform client is responsible for the unified message management platform CSP of other development teams. The background sends the instant message to the CSP platform. The CSP platform is put into the message queue and passes the GCM/FCM (Android platform) and APNS according to the different platforms of the device. (iOS platform) and the client periodically post messages to the server on the server side (Windows platform). Synchronization of various control rules is triggered by message notifications and synchronized via HTTP protocol (not required for iOS, iOS is synchronized via Apple's MDM protocol).

- The background uses OAuth 2 as the authorization mechanism, and through the account system integrated with the unified account management platform of other development teams as the authentication mechanism, it does not store the account authentication credentials itself, and associates the account system of the unified account management platform through the email address. The family members log in to the system through the platform clients, the parents use the account system authentication of the unified account management platform, and the children use the password authentication composed of multiple strings that are not repeated by the parents.

- iOS platform application control and device time control through Apple's MDM protocol. The Android platform application control and device can be controlled by the Android client. The background scheduling task (Scheduler) periodically calculates the change of the iOS platform device control rule of the child in the family member and delivers the command to be applied to the background MDM service command queue, and the background MDM service periodically checks the command queue and controls the rule. The changed commands are finally sent to the target device according to Apple's MDM protocol.

- Back-end integration with other third-party services, including: a unified account management platform for verifying home accounts (see above); an iTunes service for getting application metadata for the iOS platform; an application for getting the Android platform (and some iOS apps that are not available through the iTunes service) metadata provider; CSP platform for message notifications (see above); Google Geocoding API service (later disabled, instead acquired on the client side Directly reported to the back-end server); Apple APNS service for integrating Apple's MDM protocol; SMTP service for sending mail to new registered accounts and newly added children.

- The background data store uses Cassandra clusters in a unified manner. All services are connected to the same Cassandra cluster. Each service has its own one or more Cassandra Keyspaces for storing data. Use the Cassandra Python client provided by Datastax and add asynchronous support for Tornado to manipulate the database using CQL. The application metadata of each platform is stored with the TTL attribute, which is implemented as a caching mechanism, so that the application metadata can be updated in time.

- The organization of the background code uses the standard Python package. It consists of multiple Python packages. Each service has a corresponding Python package. The general code is provided by one of the Python packages, which encapsulates the basic operations and services of the database. A unified call base for inter-interfaces, a common Health interface for each service, and a unified command line foundation for each service. The entry points of the Python package and the stevedore implementation plugin mechanism of OpenStack are used multiple times in Python code.
- The background code guarantees code quality using a large number of unit tests and static code analysis tools. Each new code change must ensure that the unit test coverage is over 80%, otherwise it will not pass, and the static code analysis tool uses flake8 and partially adjusted pylint, and also uses the code complexity Radon to ensure the complexity of the code. And maintainability. These are enforced in the integration test, along with the code review system Gerrit to prevent unconfirmed changes from being submitted directly to the code base.
- The background also provides a script for collecting periodic data and sending results. It connects to the database every week to collect all account information (including family member roles, etc.) and all device information (including device platform, device registration time and last online time). Etc.) Finally, the collected results are sent to the appropriate personnel by means of email attachments.
- Logstash is deployed on each machine (or AWS EC2 instance) in the background. DevOps uses the ELK technology stack to collect management logs, then checks the logs through Kibana, and also configures AWS's CloudWatch to view the logs.
- Back-end use of external monitoring services to regularly monitor the health of each service (the Health interface exposed through each service) from all regions of the world.
- The integrated test environment is performed in the TeamCity host and Agent machines in the compute center. The test environment for QA engineers is deployed on multiple EC2 examples from AWS. The pre-release (Staging) environment and production environment were also initially deployed in AWS's EC2 example and later migrated to Intel's own computing center.
- Later, Verizon joined the integration of third-party collaboration.

Technology stack:

- Server backend: Python, Tornado, Cassandra, nginx, Supervisor, Ubuntu Server, etc.

Links:

- Safe Family: http://family.mcafee.com/