HelixOTA

HelixOTA

به‌روزرسانی‌های بی‌سیم جهانی و کاملاً جداشده — طراحی بدون آجر شدن

خلاصه

Helix OTA یک سیستم به‌روزرسانی بی‌سیم جهانی و عمیقاً جداشده است: یک صف کنترل Go به همراه عامل‌های کلاینتی برای هر سیستم‌عامل، طراحی‌شده برای ارائه به‌روزرسانی‌های ایمن و و اپلیکیشن به ناوگانی از یک برد واحد تا میلیون‌ها دستگاه. اولین هدف آن firmware مرحله‌ای است Orange Pi 5 Max روی اندروید ۱۵.

توضیح کوتاه

به همراه Go یک سیستم به‌روزرسانی بی‌سیم جهانی است — یک صف کنترل Helix OTA عامل‌های کلاینتی برای هر سیستم‌عامل — که برای جلوگیری از خرابی سیستم، اعتبارسنجی آپلودها و Orange Pi 5 Max انتشار مرحله‌ای دقیق طراحی شده است. اولین هدف آن اندروید ۱۵ روی و ویندوز در برنامه‌های آینده قرار دارند Linux است و آداپتورهای.

توضیح مفصل

جهانی، عمومی و عمیقاً جداشده است که بر (OTA) یک سیستم به‌روزرسانی بی‌سیم Helix OTA پایه یک وعده بی‌چون‌وچرا ساخته شده است: یک به‌روزرسانی هرگز نباید دستگاه سالم‌را به آجر تبدیل کلاینتی برای هر **ها/عامل‌هایSDK**، Go سروری **صف کنترل** کند. این سیستم شامل یک سیستم‌عاملی از طریق هر مدیریتی است و از پایه برای قابلیت تعبیه در **داشبورد** سیستم‌عامل و یک آداپتورهای قابل اتصال طراحی شده است، نه بازنویسی از ابتدا برای هر پلتفرم. اولین هدف پیاده‌سازی، است، جایی که خط تولید، تصاویر فلش‌را در Orange Pi 5 Max اندروید ۱۵ (همه نسخه‌ها) روی اعتبارسنجی‌شده و فایل‌های هش اجباری تولید می‌کند، به طوری که هیچ OTA کنار یک فایل فشرده ویندوز و سایر سیستم‌عامل‌ها در Linux، مصنوعی بدون اثر انگشت قابل تأیید به دستگاه نمی‌رسد؛ نقشه‌راه، پشت همان درز آداپتور قرار دارند و تنها منتظر آداپتور خود هستند — نه بازنویسی.

طراحی این سیستم بر اساس تضمین‌های سخت‌گیرانه‌ای است که توسط اپراتور تعیین شده و به عنوان ثابت‌های غیرقابل مذاکره معماری در نظر گرفته می‌شوند: صفر خرابی سیستم، اعتبارسنجی اجباری هر مصنوع پیش از استقرار، انتشار مرحله‌ای دقیق (همه‌جا یا مرحله‌ای با گام‌های ۵/۱۰/۳۰...۱۰۰٪ و قابلیت توقف و پیشروی)، مشاهده‌پذیری کامل ناوگان و مقیاس‌پذیری خطی از یک برد روی میز تا — سمت دستگاه A/B میلیون‌ها دستگاه در میدان. معماری قفل‌شده، به‌روزرسانی‌های بومی اندروید

و بازگشت خودکار در صورت خرابی بوت — را AVB/dm-verity با AOSP update_engine سفارشی و جداشده جفت می‌کند، به طوری که ایمنی هم در مسیر بوت نزدیک Go با یک صف کنترل هم در سرور وجود دارد، نه در یک لایه شکننده. دو درز به عمد قابل استخراج نگه داشته و به سخت‌افزار شده‌اند: درز آداپتور سیستم‌عامل که وعده جهانی‌بودن واقعی را به همراه دارد، و درز موتور انتشار که کمپین‌های مرحله‌ای را مستقل از سیستم‌عامل می‌سازد. کل سیستم به شش زیرماژول عمومی و مستقل تجزیه شده است — بلوک‌های سازنده قابل استفاده مجدد به جای یک ota-* نسخه‌بندی‌شده مونولیت.

در حال حاضر در مرحله تدوین مشخصات/تحقیق و توسعه پوشش تست قرار دارد؛ Helix OTA مخزن حاوی پیکره طراحی معتبر، خط لوله صادرات مستندات و داربست زیرماژول‌ها است و به صراحت مطابق با حاکمیت ضد-لافزنی — اعلام می‌کند که یک سرور و عامل تولیدی نهایی هنوز وجود — ندارد. آنچه امروز عرضه می‌شود، طرح کلی و داربست آن است که صادقانه به این صورت برچسب‌گذاری شده است.

محتوا

چرا آن‌را ساختیم

معمولاً برای هر دستگاه و هر سیستم‌عامل از نو اختراع می‌شود و یک به‌روزرسانی بد می‌تواند کل OTA برای این ساخته شد که یک سیستم به‌روزرسانی جهانی و Helix OTA. ناوگان‌را از کار بیندازد ایمنی‌محور باشد که هر سیستم‌عاملی می‌تواند از طریق آداپتورها آن‌را به کار گیرد؛ با تضمین‌های بازگشت به عقب و اعتبارسنجی که در معماری تعبیه شده‌اند، نه اینکه بعداً به آن اضافه شوند.

چرا انقلابی است

«این سیستم از پذیرفتن «هرگز دستگاه‌را از کار نینداختن» و «به‌روزرسانی تدریجی و قابل‌مشاهده به‌عنوان ویژگی‌هایی که امیدوارید تحت فشار کار کنند، خودداری می‌کند — این‌ها اصول ساختاری غیرقابل‌تغییری هستند که هم در مسیر بوت و هم در سطح کنترل‌پلن جای گرفته‌اند. همچنین با تبدیل موتور به‌روزرسانی و لایه سیستم‌عامل به درزهای قابل‌تعویض به‌جای فرضیات ثابت، همان کنترل‌پلن می‌تواند امروز اندرویدرا هدایت کند و آماده باشد تا بعدها سیستم‌عامل‌های دیگررا نیز تنها با افزودن یک آداپتور مدیریت کند — بدون انشعاب، بازنویسی یا اختراع مجدد تضمین‌های ایمنی که از قبل به آن‌ها اعتماد دارید.

نوآوری‌ها

- **دو درز قابل‌استخراج** — یک درز آداپتور سیستم‌عامل و یک موتور به‌روزرسانی مستقل از سیستم‌عامل — که «جهانی بودن»را از یک واژه تبلیغاتی به ویژگی ساختاری کدبیس تبدیل می‌کند.
- **ایمنی عمیق چندلایه**: A/B بومی سمت دستگاه (update_engine) + AVB/dm-verity + اعتبارسنجی سمت سرور لایه‌بندی روی بازگشت خودکار در صورت شکست بوت، که شده‌اند — یک به‌روزرسانی باید از چندین دروازه مستقل عبور کند تا بتواند ماندگار شود.
- **تجزیه کاتالوگ‌محور و غیرهمبسته** — قابل‌استفاده و مستقل از نسخه که ota-* به شش زیرماژول می‌توانید به‌صورت انتخابی از آن‌ها استفاده کنید، نه اینکه مجبور باشید یک کل یکپارچه‌را ببلعید.
- **انتقال اولیه HTTP/3 (QUIC)** با بازگشت خودکار به HTTP/2 و فشرده‌سازی قابل‌توافق Brotli/gzip — تحویل مدرن و کم‌تأخیر که به‌جای شکست، به‌صورت تدریجی تنزل می‌یابد.

- **مهندسی ضدفریب:** طراحی و وضعیت به‌صراحت به‌عنوان فاز مشخصات علامت‌گذاری شده‌اند و هیچ چیز نساخته‌شده‌ای هرگز به‌عنوان عرضه‌شده ادعا نمی‌شود — صداقت به‌عنوان یک ارزش مهندسی درجه‌یک اعمال می‌شود، نه یک سلب‌مسئولیت در پاورقی.

بزرگ‌ترین چالش‌های فنی و راه‌حل‌های ما

- **تضمین اینکه یک به‌روزرسانی بد هرگز دستگاه‌را از کار نیندازد** — سخت‌ترین وعده در OTA. این چالش با الزام به A/B حل شد: بومی سمت دستگاه اندروید در اسلات `update_engine` غیرفعال می‌نویسد در حالی که اسلات فعال به کار خود ادامه می‌دهد، AVB/dm-verity زنجیره بوت‌را به‌صورت رمزنگاری‌شده تأیید می‌کند، و اگر اسلات جدید نتواند بوت شود، دستگاه به‌طور خودکار به عقب بازمی‌گردد — همه این‌ها با اعتبارسنجی اجباری پیش از استقرار پشتیبانی می‌شوند تا یک محموله خراب پیش از ترک سرور شناسایی شود.
- **یک سیستم، سیستم‌عامل‌های متعدد** — با امتناع از گنجاندن فرضیات اندروید در هسته اصلی حل شد. یک درز آداپتور قابل‌تعویض سیستم‌عامل، جزئیات پلتفرم‌را جدا می‌کند و یک درز موتور به‌روزرسانی مستقل از سیستم‌عامل، منطق کمپین‌را قابل‌انتقال نگه می‌دارد؛ هر کدام به‌عنوان زیرماژولی جداگانه حفظ می‌شوند تا افزودن یک سیستم‌عامل جدید همیشه یک افزونه باشد، نه جراحی روی کل سیستم.
- **به‌روزرسانی‌های مرحله‌ای و قابل‌توقف** — با یک موتور به‌روزرسانی اختصاصی حل شد که بر اساس گروه‌های درصدی با آستانه‌های موفقیت/خطا و کنترل صریح توقف/پیشروی عمل می‌کند، و جدا نگه داشته شده تا همان موتور بتواند کمپین‌ها‌را مستقل از HTTP عمداً از وابستگی به انتقال هدایت کند.

محتوا

پشتهٔ فناوری

- **Go + Gin** — به‌دلیل مدل همزمانی و ردپای استقرار سبک انتخاب شده است؛ کنترل‌پلن REST موتور انتشار و اعتبارسنجی‌های مصنوعات‌را قدرت می‌بخشد و سطح اصلی / با آدرس `api/v1` را در معرض دسترسی قرار می‌دهد.
- **Kotlin/KMP** — اندروید روی دستگاه بتواند OTA به این دلیل انتخاب شده که عامل منطق‌را بین اهداف مختلف به اشتراک بگذارد؛ مالک چرخهٔ کامل دستگاه شامل پرس‌وجو، دانلود، تأیید، اعمال و گزارش‌دهی است.
- **HTTP/3 (QUIC) → HTTP/2** — QUIC به‌عنوان حمل‌ونقل اصلی برای تحویل کم‌تأخیر و مقاوم در برابر اتصالات موبایلی ناپایدار انتخاب شده است، با قابلیت بازگشت خودکار به HTTP/2 به‌صورت **Brotli/gzip** تا هیچ دستگاهی بدون پشتیبانی نماند؛ فشرده‌سازی درخواست‌به‌درخواست مذاکره می‌شود تا حجم محموله‌ها کاهش یابد.
- **PostgreSQL** — به‌دلیل حفظ یکپارچگی‌رابطه‌ای بین رجیستری دستگاه‌ها، کمپین‌ها و تله‌متری انتخاب شده است، جایی که صحت وضعیت ناوگان اهمیت بیشتری از سرعت خام نوشتن دارد.
- **MinIO / S3** — به‌عنوان مخزن مصنوعات انتخاب شده تا تصاویر بزرگ فریم‌ور در فضای ذخیره‌سازی اشیاء کالایی قرار گیرند و از لایهٔ‌رابطه‌ای جدا شوند.
- **AOSP `update_engine` + AVB/dm-verity + `boot_control`** — به این دلیل انتخاب شده که استفاده مجدد از مکانیزم‌های تست‌شدهٔ اندروید برای به‌روزرسانی مجازی A/B و بوت تأییدشده امن‌تر از اختراع یک به‌روزرسان سفارشی است؛ برای مدیریت تعویض اسلات‌ها و تأیید بوت‌رمزنگاری‌شده روی دستگاه استفاده می‌شود.

- **React** — برای داشبورد مدیریتی انتخاب شده است که اپراتورها از طریق آن وارد می‌شوند، مصنوعات‌را آپلود می‌کنند، انتشارهارا مدیریت می‌کنند و سلامت ناوگان‌را در یک مکان متمرکز رصد می‌کنند.
- **OpenTelemetry + Prometheus/Grafana** — به‌عنوان ابزار اندازه‌گیری بی‌طرف نسبت به فروشنده انتخاب شده است؛ برای قابل‌مشاهده‌کردن هر مرحله از انتشار در معیارها و داشبوردها به‌کار می‌رود تا نیازی به حدس‌وگمان نباشد.

وضعیت و یادداشت‌های شفافیت

- **وضعیت: در حال توسعه.** مطابق با اصول ضداغراق پروژه، **هیچ سرور یا عامل عملیاتی آماده‌ای** این مرحله، فاز مشخصات/تحقیق و توسعهٔ پوشش تست است. مخزن شامل — **هنوز وجود ندارد** پیکرهٔ معتبر طراحی، خط لولهٔ صادرات مستندات و چارچوب زیرماژول‌ها است.
- شش زیرماژول عمومی قابل‌استفاده مجدد (`ota-protocol`، `ota-artifact-validator`، `ota-rollout-engine`، `ota-update-engine-bridge`، `ota-android-agent`، `ota-telemetry-schema`) تحت آدرس `github.com/HelixDevelopment/` قرار دارند.
- ارقام پوشش تست و تأخیر در مخزن، دفترچهٔ در حال پیشرفت خود پروژه هستند و به‌طور مستقل ذکر README که در فایل HelixConstitution تأیید نشده‌اند. شماره‌های بندهای هستند **تأییدنشده**، شده‌اند.
- **مجوز: Apache-2.0.**

اولویت سطح: Helix-اصلی.