Bild 1. En uppkopplad IoT-enhet, som en smart dörrklocka, är sårbar för cyberattacker. CRA kräver att tillverkaren snabbt installerar åtgärder mot nyupptäckta sårbarheter.

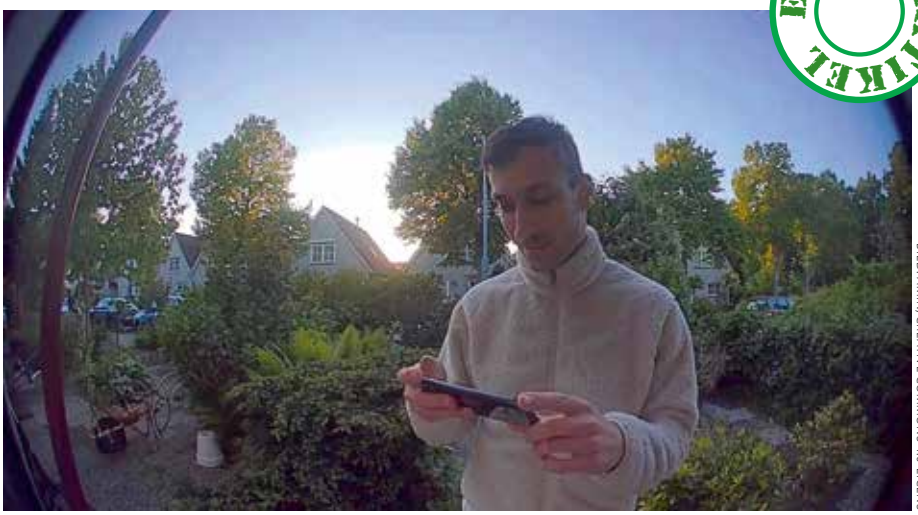

BILD: AMIN, CREATIVE COMMONS 4.0-LICENS

EXPERTARTIKEL

Av George Grey, Qualcomm

Qualcomms teknikchef **George Grey** är en av grundarna av Foundries.io, en molnbaserad plattform för utveckling och fjärradministration av IoT-enheter. Plattformen är baserad på öppen källkod, som är välbekant terräng för George Grey som även var med och grundade Linaro och Tadpole. Innan dess ledde han Javaföretaget SavaJe. George Grey tog sin examen i elektroteknik på universitetet i Cambridge.

CYBER RESILIENCE ACT: Utmaningar

Att uppfylla kraven i EU:s CRA (Cyber Resilience Act) kommer snart att vara en av de stora tekniska och organisatoriska utmaningarna för alla som marknadsför inbyggda system i Europa. Förordningen trädde i kraft den 10 december 2024 och från den 11 december 2027 är många av de viktigaste kraven bindande.

Ett av de centrala kraven är att kunna uppdatera programvaran på din produkt även efter att du sålt den, varje gång det dyker upp ett nytt cyberhot. Specifikt måste du kunna reagera på CVE-meddelanden (Common Vulnerabilities and Exposures). OEM:er känner idag till vilka typer av hårdvara som måste finnas i system för att de ska kunna CRA-godkännas (HSM, Hardware Security Module, eller TPM, Trusted Platform Module) men det är fortfarande ganska nytt och ovant hur en process för att upprätthålla programvarusäkerhet ser ut, och hur man distribuerar uppdateringar till produkter som redan är driftsatta.

Att ämnet i sig väcker stress är förståeligt – programuppdatering är komplext och svårhanterligt. På Foundries.io har vi under många år levererat teknik och support till inbyggnadsföretag inom området. Och vi menar att de krav som ställs i CRA inte är något som man behöver ligga vaken över.

Men under införandet av förordningen har vi märkt att även om det finns mycket i processen som kan automatiseras och effektiviseras, så saknar branschen fortfarande viktiga verktyg och hjälpmedel som skulle kunna göra CRA-efterlevnaden betydligt mindre betungande.

Hur CRA kräver att mjukvaruuppdateringar hanteras

Syftet med CRA är att ”skydda konsumenter och företag som köper programvaru- eller hårdvaruprodukter som innehåller en digital komponent”. EU-lagen är en reaktion på bristande cybersäkerhet i mjukvara idag och avsaknaden av teknik för snabba säkerhetsuppdateringar (se bild 1). Lagen ställer nya krav på både tillverkare och återförsäljare, i synnerhet vad gäller att kunna upprätthålla säkerhet under hela livscykeln.

Tiden är förbi då tillverkaren kunde skeppa produkten och sedan glömma att den existerade. Skyldigheten att skydda mot cyberattacker fortsätter långt efter att produkten nått slutanvändare.

CRA kodifierar skyldigheten på flera sätt. Du måste:

- åtgärda sårbarheter som identifieras i CVE-meddelanden, även efter leverans
- för varje producerad enhet måste du upp-rätthålla en SBOM (Software Bill of Materials) som kan användas för att identifiera vilka CVE-sårbarheter som är tillämpliga
- du måste åtgärda sårbarheter ”utan dröjsmål”
- regelbundet testa och granska produktsäkerhet
- ha en policy för sårbarhetsrapportering
- skyndsamt och på ett cybersäkert sätt distribuera kostnadsfria uppdateringar

OEM-tillverkaren är skyldig att dokumentera programvaran i alla enheter som lämnar fabriken. Det görs i en så kallad SBOM (Software Bill of Materials) som måste hållas uppdaterad på alla förändringar av koden. Tillverkaren måste kunna veta om någon del av kodbasen är exponerad för en känd sårbarhet och måste snabbt kunna åtgärda och distribuera en gratis lagning till alla användare (se bild 2).

Värdet av en infrastruktur för programuppdateringar

Kodbasen i moderna uppkopplade produkter, särskilt de som bygger på Linux, är komplex. Den kan bestå av hundratals eller

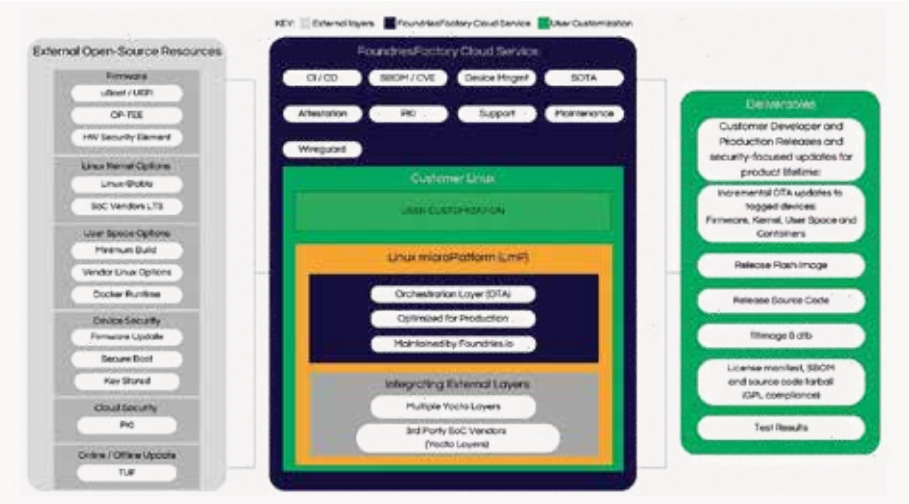

Bild 3. FoundriesFactory integrerar extern öppen källkod som stöd för utveckling, säkerhet och drift.

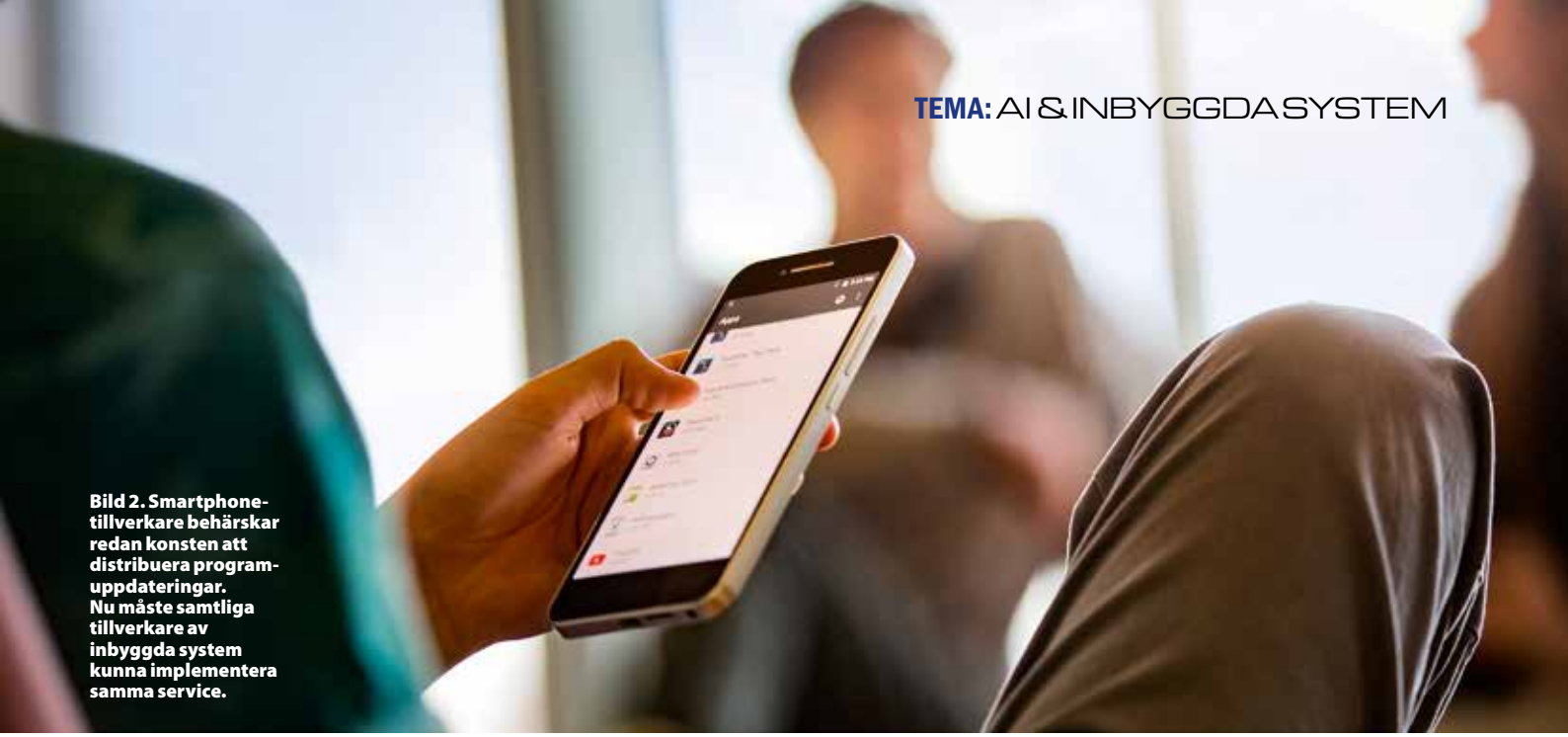

Bild 2. Smartphone-tillverkare behärskar redan konsten att distribuera program-uppdateringar. Nu måste samtliga tillverkare av inbyggda system kunna implementera samma service.

– och en öppen lösning

tusentals komponenter i en blandning av öppen källkod, kommersiell tredjepartskod och egen kod. Det är i praktiken omöjligt att manuellt hålla reda på all denna kod manuellt – dessutom för varje produktvariant. Det krävs en automatiserad SBOM-process såväl under utveckling som efter leverans.

Även hanteringen av säkerhetsfunktioner som autentisering, autentisering, kryptering och säker boot, är komplex. Produktens privata nycklar måste lagras säkert. God hantering av PKI (Public Key Infrastructure) är grundläggande i CRA.

OEM:er behöver dessutom ett system för fleet management – hantering av flottor av driftsatta enheter. Du behöver hålla reda på varje enhet med en unik identifierare, och du behöver kunna distribuera rätt uppdatering till rätt enhet vid rätt tillfälle, och på ett säkrat sätt, ända till den dag då enheten till slut tas ur drift. Detta stoppar en angripare från att kunna smuggla in en förfalskad enhet i flottan som ett sätt att komma åt känslig information eller proprietär kod.

Funktionerna kräver en systematisk hantering av loggning, lagring och datahantering, både när produkten utvecklas och tillverkas (development-fasen) och när den underhålls efter leverans (operations-fasen). Funktionerna är grundläggande i CRA och därför har OEM:er börjat införa formella ramverk för devops (development och operations). Just den typen av system är FoundriesFactory från Foundries.io ett typexempel på (se bild 3).

Ett devops-system bör omfatta följande verktyg och funktioner:

- PKI-hantering
- SBOM-generering och underhåll
- flott-administration, fleet management
- CI/CD – vid varje kodändring uppdateras och testas hela systemet automatiskt
- kod ska hanteras på ett cybersäkert sätt

Genom att använda sådana system genom hela produktlivscykeln förenklas processer som annars skulle varit komplexa, tidskrävande och felbenägna.

CRA:s sista kryssrutor

Alla utom de allra minsta OEM:erna behöver ha ett devops-ramverk för att kunna uppfylla CRA-kraven. Men det räcker inte. Andra viktiga verktyg krävs, och tyvärr finns de fortfarande inte som öppen källkod. Här är två verktyg som lyser med sin frånvaro:

1. Verktyg för CVE-analys. Dagens CVE-sökverktyg är ofta primitiva. Det beror på att de helt enkelt jämför namnet på ett programbibliotek i din programvara med namnet på ett bibliotek i en känd CVE-sårbarhet. Men att du använder ett bibliotek betyder inte att du använder den del av biblioteket som är sårbar. CVE-meddelandet gäller oftast en sårbarhet som finns i en viss del av koden – inte i hela paketet. Så resten av koden kan mycket väl vara säker.

Om din produkt enbart använder en viss del av ett programbibliotek – och inte den del som är sårbar – så finns ingen anledning att skicka ut en säkerhetsuppdatering. Men en enkelt konstruerad CVE-skannerfunktion kommer att flagga systemet som sårbart, bara för att biblioteket i sig nämns.

Det visar att det behövs mer avancerade verktyg, som kan gå in i källkoden och se om den innehåller exakt den sårbara kod som nämns i CVE:n. Det finns skäl att tro att framtida AI-modeller kommer att kunna användas som bas sådana verktyg.

2. Verktyg för att dokumentera de CRA-åtgärder du vidtar. Det andra som saknas som öppen källkod är revisionshantering. CRA föreskriver böter vid bristande efterlevnad i proportion till företagets storlek. OEM:er behöver kunna visa exakt vad de gjort – för varje enskild enhet – för att åtgärda sårbarheter. Din plattform bör alltså kunna logga vilka sårbarheter som flaggats, vilka patchar som rullats ut, och till vilka enheter.

Precis som SBOM-verktyget FoundriesFactory automatiskt skapar en lista över vilka mjukvarukomponenter som ingår i varje enhet, så måste ett bra verktyg för efterlevnadsrapportering hålla koll på vilka sårbarheter som rapporteras och vilka åtgärder som vidtagits, för varje enhet.

Förhoppningen är att verktyg av dessa slag kommer att utvecklas i form av öppen källkod och att den blir spridd och i praktiken en standard. Den dagen slipper alla OEM:er uppfinna hjulet på nytt.

Därför har FoundriesFactory valt att använda The Update Framework (TUF) som bas. Standardisering kring öppen källkod skapar en positiv spiral. Därför valde FoundriesFactory TUF (The Update Framework) för leverans och installation av säkerhetsuppdateringar. Ytterligare andra element i FoundriesFactory har en bas i öppen källkod av samma skäl.

Uppdatering är en process, inte bara ett programpaket

Kravet på CRA-efterlevnad gör att OEM:er nu tvingas ta programuppdateringar på allvar och inse att det omfattar mer än uppdateringen i sig. Det krävs ett helt ekosystem av säkerhet, övervakning, cybersäkerhet, fleet management och devops som snabbt kan svara när det dyker upp nya sårbarheter.

För många OEM:er är ett etablerat devops-ramverk rätt lösning – det är ett automatiserat sätt att hålla reda på enheter i fält och rulla ut rätt uppdatering till rätt enhet vid rätt tillfälle. ■