Engineer

Mjukvara

de program och data som en dator kör

Vad det är

Mjukvara är varje del av ett datorsystem som kan ändras genom att man skriver i stället för att man lödar: programmen, den data de bär med sig och inställningarna som talar om för dem hur de ska bete sig. Ordet används nästan alltid utan plural, i bestämd form mjukvaran, och lika ofta heter det programvara och programvaran. En enskild del kallas ett program, och något som en människa öppnar för att utföra en uppgift kallas oftast en app eller en applikation. ”Kod” brukar betyda den källkod som en utvecklare redigerar, och specifikationer förkortar ibland hela begreppet till SW.

Dess vanliga motsats är hårdvara, den fysiska maskinen, och i glappet mellan dem finns fast programvara, ofta kallad firmware. Ordlistan från NIST drar den gränsen genom förändring: mjukvara är datorprogram och tillhörande data som kan skrivas eller ändras dynamiskt medan de körs, medan firmware är den sort som inte kan ändras under körning. En bärbar dators operativsystem är mjukvara; koden i dess tangentbordsstyrning är firmware; tangentbordet är hårdvara.

Inuti mjukvaran finns lager. Systemprogram, t.ex. ett operativsystem eller en drivrutin, hanterar maskinen. Applikationsprogram gör något för en människa: ett kalkylark, en webbplats för bokningar, en meddelandeklient. Mellan dem finns bibliotek och programpaket, färdiga delar som båda sorterna sätts ihop av, och i praktiken kom det mesta av koden i en modern produkt den vägen snarare än att skrivas för just den.

Ytterligare en skillnad avgör mycket längre fram. Källkod är det som människor skriver och läser; en kompilator eller en tolk gör om den till instruktioner som en processor utför. Båda är mjukvara, men de är inte utbytbara när du köper den. En leverans av enbart kompilerade filer är något som bara leverantören kan ändra.

Använd begreppet i ett beslut

Använd ”Vad är mjukvara?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Systemets form utarbetad på papper först, så att delarna passar ihop och kostnaden är känd innan någon skriver kod.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. En infrastrukturplan för 14 avdelningar — 14 forskningsavdelningar under en och samma infrastrukturplan

En infrastrukturplan för 14 avdelningar Hör av er →

Varför det spelar roll

Den som beställer, köper eller hyr mjukvara väljer samtidigt vem som bär dess underhåll. Det som gör mjukvara attraktiv, att det inte kostar något att kopiera den och att det är billigt att ändra den efter leverans jämfört med att ändra en maskin, är också skälet till att den aldrig blir färdig. Komponenterna som den byggdes av publicerar rättelser. Operativsystem och webbläsare pensionerar gränssnitt. En sårbarhet offentliggörs i ett bibliotek som ingen i teamet minns att ha valt. Var och en av dem blir arbete för den som äger koden.

De vanliga vägarna fördelar det arbetet på olika sätt:

- **Hyrd som tjänst.** Leverantören sköter uppdateringarna och servrarna. Du ärver leverantörens releaseschema, prissättning och besväret med att lämna, och din data ligger på leverantörens sida.
- **Licensierad och driven av dig själv.** Du väljer när du uppgraderar, och du gör också uppgraderingarna, säkerhetskopieringen och säkerhetsuppdateringarna.
- **Byggd för dig.** Den passar processen i stället för tvärtom, och budgeten för underhållet är din så länge den körs. Kostnaden som avgör om det var klokt kommer efter lanseringen.
- **Öppen källkod.** Under en licens som uppfyller Open Source Definition får du använda, studera, ändra och sprida den vidare. Någon måste ändå installera uppdateringarna; att något är gratis att skaffa är en annan egenskap än att det är gratis att driva.

Vilken väg du än väljer är två frågor om resultatet värda att ställa innan du skriver på. Vad får du vid slutet: källkoden, instruktionerna för att bygga den och en förteckning över vad som finns i den? Och hur ska leverantören hålla den säker? Secure Software Development Framework från NIST finns delvis för det andra samtalet, eftersom det ger köpare och leverantörer ett gemensamt språk för utvecklingsmetoder. För det första samtalet definierar en rapport från NTIA en

materialförteckning för mjukvara, på engelska software bill of materials eller SBOM, som en formell förteckning över detaljerna i, och leveranskedjans relationer mellan, de komponenter som används för att bygga mjukvaran. Att be om en är ett billigt sätt att se hur mycket av det du betalar för som har skrivits av någon annan.

Så ser det ut i praktiken

Att veta vad som levererades är den del som team hoppar över. På en plattform flyttades säkerhetsarbetet från åsikter in i bygget: i ett kontinuerligt säkerhetsprogram genereras en materialförteckning för programvaran vid varje bygge med Anchore och Syft, Dependabot bevakar 6 ekosystem, CodeQL och ZAP analyserar koden och den körande instansen, och de actions från tredje part som bygget använder pekas ut med en commit-hash, aldrig med en tagg som kan flyttas. Vid sidan av automatiseringen finns 21 skriftliga säkerhetsgranskningar, bland dem en hotmodell. En sårbarhet som de konfigurerade kontrollerna känner igen kan få bygget att misslyckas. Sidan säger också rakt ut vad det kostar: mängden fynd kräver en triagering som aldrig tar slut.

Att få mjukvara från en ändring till produktion är den andra halvan. Där releaser hade byggt på att någon kom ihåg en följd av manuella steg, och gjorde dem lite olika varje gång, sparade automatiserade pipelines för bygge, test och driftsättning uppskattningsvis 4 000 timmar och gjorde releaserna snabbare och mer tillförlitliga. Vinsten kom av att vägen var identisk varje gång, inte av att någon var mer försiktig.

Gränsen mellan mjukvara och hårdvara är dessutom högst praktisk på en arbetsbänk för reparationer. Reparationer av bärbara datorer omfattade skärmar, gångjärn, tangentbord och moderkort, och mjukvaruproblem i Linux, Windows och Mac. En bärbar dator som inte startar kan bero på laddaren, kortet eller batteriet, så arbetet började med att hitta det verkliga felet innan något rördes.

Var det går fel

En säkerhetsvarning blir en veckas utgrävning

En sårbarhet offentliggörs i en spridd komponent, och ingen kan säga vilka av organisationens system som innehåller den. Förteckningen har bara funnits i folks huvuden och i låsfiler som ingen läser, så den måste återskapas under press,

system för system. Generera en materialförteckning som en del av varje bygge och spara den tillsammans med den release som den beskriver. Frågan ”berörs vi?” blir då en sökning.

Releaser beror på vem som gör dem

Driftsättningen fungerar när en person kör den och går sönder när en annan gör det, och skillnaden visar sig vara ett steg som bara finns i någons minne. Varje release blir en liten improvisation. Lägg hela vägen i en pipeline som bygger, testar och driftsätter på samma sätt varje gång, och behandla varje manuellt steg som finns kvar som en defekt att ta bort.

Budgeten slutar vid lanseringen

Två eller tre år senare känns varje uppdatering riskabel, ramverket ligger flera huvudversioner efter, och det enda förslaget på bordet är att skriva om allt. Mjukvaran betalades som en byggnad och lämnades sedan i fred, medan allt runt omkring den fortsatte att röra sig. Avsätt en stående budget för underhåll, uppgradera beroenden i små regelbundna steg medan varje steg fortfarande är litet, och räkna med att plattformarna under den ändras i sin egen takt.

Ett namn står still medan koden bakom det ändras

Samma bygge ger olika resultat på två olika dagar utan någon commit emellan. Någonstans pekar en referens på ett rörligt mål: ett versionsintervall, en `latest`-tagg, en CI-action som refereras med en tagg som ägaren kan flytta. Lås kod från tredje part till en exakt version eller, där taggar kan flyttas, till en commit-hash, committa låsfilen och låt en bot föreslå uppdateringarna, så att varje ändring kommer fram som en diff som går att granska.

Läs mer

- NIST CSRC Glossary: software — definitionerna som publikationerna från NIST använder, bl.a. den som skiljer mjukvara från firmware.
- NIST SP 800-218, Secure Software Development Framework — en kärna av metoder för att bygga mjukvara med färre sårbarheter, skriven så att köpare kan använda den med sina leverantörer.
- NTIA: The Minimum Elements for a Software Bill of Materials — vad en förteckning över komponenterna i mjukvara måste innehålla för att vara till nytta.
- The Open Source Definition — de tio kriterier som en licens måste uppfylla innan mjukvara som släpps under den räknas som öppen källkod.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är mjukvara?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/software/