Engineer

Systemkjerne

kjernen i et operativsystem

Hva det er

En systemkjerne, på engelsk kernel, er programmet i midten av et operativsystem som kjører med prosessorens fulle privilegier og styrer tilgangen til maskinvaren. Den bestemmer hvilket program som får kjøre neste gang, deler ut minne og holder hvert programs minne adskilt fra resten, leser og skriver disker gjennom filsystemer, flytter pakker gjennom nettverksstakken og håndhever reglene for hvem som får tilgang til hvilken fil, nettverksport eller enhet. Alt annet kjører i brukerrommet, user space, med færre privilegier og ber kjernen om å handle på sine vegne.

Disse forespørslene er systemkall, og Linux-manualen beskriver dem som det grunnleggende grensesnittet mellom en applikasjon og Linux-kjernen. Programmer gjør dem sjelden for hånd. De kaller innpakninger i et C-bibliotek som glibc, som sender argumentene videre og gjør kjernens svar om til en returverdi og en feilkode. Når et program stopper med «Permission denied», er den ordlyden som regel kjernens svar på et slikt kall, oversatt av biblioteket.

Kjerner skiller seg i hvor mye de holder inne i seg selv. Linux har drivere, filsystemer og nettverksstakken i selve kjernen og kan laste inn mer som moduler mens den kjører, en utforming som vanligvis kalles monolittisk. Hver driver kjører da med kjernens fulle privilegier. En mikrokjerne beholder bare det minste nødvendige. seL4 leverer mekanismer for å styre tilgangen til det fysiske adresserommet, avbrudd og prosessortid, og kjører drivere, filsystemer og nettverksstakker som vanlige programmer i brukermodus. Den verifiserte 64-biters RISC-V-versjonen er på rundt 10 000 linjer kode, med et formelt bevis for at C-implementasjonen gjør det spesifikasjonen sier. Kjernen i macOS, XNU, er en hybrid som kombinerer mikrokjernen Mach med deler av FreeBSD.

En kjerneoppdatering trer normalt i kraft først når maskinen starter opp med den nye kjernen, og kjernens versjonsnummer, som 6.12, er noe annet enn utgivelsesnummeret til distribusjonen rundt den.

Bruk begrepet i en beslutning

Bruk «Hva er en systemkjerne?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Daglig ettersyn av servere, e-post, kontoer og oppdateringer, slik at det alle er avhengige av, forblir kjedelig.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Drev 10 PostgreSQL- og MS SQL-servere på Ubuntu — 10 databaseservere produksjonssatt og holdt i god stand

Drev 10 PostgreSQL- og MS SQL-servere på Ubuntu Ta kontakt →

Hvorfor det er viktig

For en virksomhet avgjør systemkjernen tre spørsmål: hvor langt fra hverandre arbeidslastene egentlig er, hva en sikkerhetsrettelse koster i nedetid, og hvor sannheten om en maskin er lagret.

Først isolasjonen. En container er et sett vanlige prosesser som kjernen har plassert i navnerom, namespaces, som gir hver gruppe sin egen visning av prosess-ID-er, nettverksgrensesnitt, monteringspunkter og brukere, og i kontrollgrupper, cgroups, som fordeler prosessortid, minne og lese- og skriveoperasjoner mot disk. Hver container på en vert kjører på vertens ene kjerne, så en svakhet i den kjernen er en svakhet under alle samtidig, og et image som har med seg sine egne biblioteker, er fortsatt avhengig av de kjernefunksjonene verten har. En virtuell maskin starter opp en egen kjerne. For tjenestene til ett team er containere en rimelig grense. For kunder som ikke stoler på hverandre, bør hver av dem få sin egen kjerne.

Deretter rettelsene. En oppdatert kjernepakke på disken endrer ingenting før maskinen starter opp med den, og på en server er den omstarten prisen. Livepatching finnes fordi brukere, med kjernedokumentasjonens ord, er motvillige til å starte på nytt: mekanismen leder enkeltfunksjoner over til rettede versjoner mens systemet kjører. Bare funksjoner kjernen kan spore, kan rettes på den måten, så livepatching kjøper tid for en hasterettelse og erstatter ikke omstarter. En plan med en fast, innøvd omstart koster mindre enn en som antar at omstarten aldri trenger å skje.

Til slutt holder kjernen den aktuelle tilstanden. En brannmurfil beskriver regler, men det er reglene i kjernens pakkefilter som avgjør hva som skjer med en pakke. En fil i `/etc/sysctl.d` beskriver en innstilling, men det er verdien under `/proc/sys` som gjelder. En tjeneste kan melde seg frisk uansett om handlingen dens noen gang nådde kjernen. Sjekker som leser kjernens egen tilstand, fanger feil som en gjennomgang av konfigurasjonsfiler ville sluppet gjennom.

Slik ser det ut i praksis

En herdingsgjennomgang av en produksjonsvert i august 2026 spurte om beskyttelsene mot brute force-angrep på SSH virket. Hver av dem var konfigurert og dukket opp i hver rapport, så gjennomgangen sjekket dem mot det kjernen faktisk gjør med en pakke i stedet for mot filene som beskriver hva som burde skje. Alle tre hadde vært virkningsløse siden verten ble bygd. Jailens blokkeringsregler gjaldt port 22, mens SSH var flyttet til port 1986. Brannmurens regler evalueres ovenfra og ned, og første treff vinner, så ratebegrensningen sto under en bredere regel som tok pakkene før den. Utestengingshandlingen kalte et pakkefilterprogram som pakkesystemet bare anbefaler, så jailen startet, talte og avgjorde, og feilet så i øyeblikket den prøvde å stenge ute. Ut av det kom to regler for repoet. Den ene krever en maskinell kontroll av hver sikkerhetsmekanisme fremfor bare en kommentar, og den andre sier at en brannmur sjekkes ut fra plasseringen av reglene i stedet for ut fra at de finnes.

Utestengingsveien fikk deretter en test som kjører hver gang konfigurasjonen blir brukt. Hver kjøring stenger ute en adresse fra en blokk som er reservert for dokumentasjon, leser den resulterende regelen tilbake fra kjernens pakkefilter og opphever utestengingen i et oppryddingssteg som kjører uansett utfall. Tilbakelesingen er det som gjør det til en test: tjenestens egen status ville meldt suksess for en utestenging som ikke ga noen regel. En reservert adresse tilhører ikke noe virkelig nettverk, så en utestenging som blir liggende etter en mislykket kjøring, stenger ingen ute. Kostnaden er noen sekunder per kjøring, og en brannmurregel som skrives til produksjon og trekkes tilbake hver gang.

En vert på 464 MB uten plass til en metrikkstack fikk sju roller som bare leser, og som gjengir maskinens aktuelle tilstand som Markdown. Øyeblikksbildet av fakta beregner prosessorutnyttelsen fra to målinger av kjernens egen teller. Sikkerhetsrapporten skriver ut brannmurreglene i den rekkefølgen de treffer, ved siden av de aktive utestengingsreglene, og regelen for alle sju rollene er at ingen tall skrives ut som kjøringen ikke har målt. De rapportene avdekket to døde sikkerhetstiltak og en lyttende nettverksport ingen hadde deklarert.

Hvor det går galt

En kjerneinnstilling som er endret for hånd, går tilbake etter neste omstart. `sysctl -w` skriver verdien bare inn i den kjørende kjernen, så den varer til omstart. Den motsatte glippen: en fil under `/etc/sysctl.d` leses ved oppstart, så endringen venter, uten at noen får vite det, på en senere omstart. Skriv innstillingen til en fil, last den inn nå, og les den gjeldende verdien tilbake:

```
echo 'vm.swappiness = 10' | sudo tee /etc/sysctl.d/90-local.conf
sudo sysctl --system
sysctl vm.swappiness
```

En CPU-graf ligger flat gjennom en lasttopp. Linjen `cpu` i `/proc/stat` teller tid siden oppstart, i enheter av USER_HZ (hundredels sekunder på de fleste arkitekturer), så én enkelt avlesning gir gjennomsnittet siden oppstart, og det rører seg knapt på en maskin som har vært oppe i måneder. Les den to ganger og del forskjellene:

```
head -1 /proc/stat; sleep 5; head -1 /proc/stat
```

Opptatt tid er endringen i user, nice, system, irq og softirq, ledig tid er endringen i idle og iowait, og utnyttelsen er opptatt tid delt på opptatt pluss ledig. Manualen kaller iowait upålitelig, så behandle den som et hint og ikke som grunnlag for et driftsvarsel.

En krasjrapport kommer tilbake med en anmodning om å gjenskape feilen på en kjerne som ikke er tainted. Kjernen merker seg selv som tainted, besmittet, når noe skjer som kan få betydning for en senere undersøkelse, som at en proprietær modul, en modul utenfor kildetreet eller en usignert modul lastes inn, og dokumentasjonen advarer om at utviklere ofte ignorerer feilrapporter fra slike kjerner. Merket blir stående til omstart, også etter at modulen er lastet ut. Les det før du sender inn rapporten:

```
cat /proc/sys/kernel/tainted
```

Null betyr at kjernen ikke er tainted. Ethvert annet tall er en sum av flagg, blant dem `1` for en proprietær modul, `4096` for en modul utenfor kildetreet og `8192` for en usignert modul. Gjenskap feilen etter å ha startet opp uten modulen som satte flagget.

Et nytt nettverkskort dukker opp i maskinvarelisten, og systemet tilbyr ikke noe grensesnitt for det. Drivere er en del av kjernen, og en kjerne som er eldre enn maskinvaren, kan mangle en. `lspci -k` viser driveren hver enhet bruker, og hvilke moduler som kan håndtere den; en enhet uten en linje for driver i bruk har ingen driver knyttet til seg. Installer en nyere kjerneserie fra distribusjonen i stedet for å bygge leverandørens modul for hånd: en modul utenfor kildetreet må bygges på nytt for hver nye kjerne, og når den lastes inn, blir kjernen tainted.

Les mer

- The Linux kernel user's and administrator's guide — kjerneprosjektets egen håndbok for å kjøre Linux.
- syscalls(2) — hvert systemkall i Linux, med kjerneversjonen som innførte det.
- namespaces(7) — de ulike typene navnerom og hvordan containere bygges av dem.
- proc_stat(5) — hvert felt i `/proc/stat`, advarselen om iowait inkludert.
- Tainted kernels — hvert taint-flagg og hvordan tallet dekodes.
- seL4 FAQ — hva en mikrokjerne utelater, og antakelsene bak beviset for seL4.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er en systemkjerne?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/kernel/