Säkra leveranskedjor för IT-system

Daniel Eidenskog, Caroline Bildsten, Bodo Endres

FOI-R--4851--SE

December 2019

FOI

**Daniel Eidenskog, Caroline Bildsten,
Bodo Endres**

Säkra leveranskedjor för IT-system

Titel	Säkra leveranskedjor för IT-system
Title	Secure cyber supply chains
Rapportnr/Report no	FOI-R--4851--SE
Månad/Month	December
Utgivningsår/Year	2019
Antal sidor/Pages	58
ISSN	1650-1942
Kund/Customer	Försvarsmakten
Forskningsområde	Informationssäkerhet
FoT-område	Operationer i cyberdomäner
Projektnr/Project no	E72777
Godkänd av/Approved by	Christian Jönsson
Ansvarig avdelning	Ledningssystem

Bild/Cover: Shutterstock, JLO-Foto

Sammanfattning

Denna studie undersöker vad cyberleveranskedjor är och vilka hotbilder som finns mot dessa. Syftet med studien är att förmedla kunskap gällande de risker som kan uppstå i cyberleveranskedjor och vad som går att göra för att motverka dessa risker. Kunskapen är avsedd att underlätta diskussioner kring cyberleveranskedjor och de risker som de medför för IT-system i Försvarsmakten. Rapporten riktar sig huvudsakligen till personer som arbetar med anskaffning, utveckling och förvaltning av Försvarsmaktens IT-system.

Inom studien genomfördes två fallstudier för att belysa komplexiteten i cyberleveranskedjor för både hårdvara och mjukvara. Fallstudiernas resultat visar att många olika aktörer är inblandade i cyberleveranskedjorna och att de är globala i sin omfattning.

Studien visar att det inte finns någon enkel lösning för att säkra upp cyberleveranskedjor. I stället krävs en gedigen riskhantering som tar hänsyn till hela kedjan. Målet med riskhanteringen är att få en resiliens mot antagonistiska hot i cyberleveranskedjan. Studien presenterar femton viktiga principer med konkreta handlingspunkter som är sammanställda från publikationer av Nist, Enisa, Mitre och Safecode. Principerna utgör ingen fullständig lista över vad som behöver utföras, men ger en utgångspunkt med några aktiviteter som är särskilt viktiga att beakta.

Nyckelord: Cyberleveranskedja, leveranskedja, leverantörskedja, resiliens, riskhantering, komplexitet

Summary

This study investigates what cyber supply chains are and investigates the threat landscape surrounding them. The purpose of the study is to give a better understanding of the risks that can emerge in a cyber supply chain for IT systems and how to mitigate them. The knowledge presented in this report is intended to ease discussions about the cyber supply chains and their associated risks for IT systems in the Swedish Armed Forces. The report foremost addresses readers working with acquisition, development and maintenance of IT systems at the Swedish Armed Forces.

Two case studies were conducted to investigate the complexity in cyber supply chains for both hardware and software. The results of the case studies show that there are many parties involved in the cyber supply chains and that they are include actors from across the globe.

The study shows that there is no simple solution for securing cyber supply chains. Instead, it requires solid risk management that addresses all suppliers, integrators, objects and processes in the supply chain. The risk management goal is to receive a resilience against adversarial threats in the supply chain. The study presents fifteen important principles with actions, based on publications from Nist, Enisa, Mitre and Safecode. These principles does not constitute a complete list with all actions needed to secure a supply chain. Instead, it provides a starting point with some actions that are especially worth considering.

Keywords: Cyber supply chain, supply chain, resilience, risk management, complexity

Innehållsförteckning

1. Inledning

Många organisationer är beroende av IT-system i sådan grad att det kan vara svårt att upprätthålla verksamheten om IT-systemen inte fungerar. Inom sjukvården kan tillgängliga och korrekta digitala journaler vara livsavgörande, inom industrin är pålitlig styrning av processer ofta centralt och inom den militära sektorn är ofta ledningssystem, sensorsystem och underrättelsehanteringssystem kritiska för verksamheten.

Militära system har ofta behov av att upprätthålla flera IT-säkerhetsaspekter samtidigt som hoten kommer från mycket kompetenta och starka antagonister med potentiellt stor förmåga att genomföra riktade angrepp. Detta kan illustreras med ett stridsledningssystem som generellt sett måste ha god *tillgänglighet* till informationen – användarna behöver komma åt relevant information när så behövs för att kunna fatta snabba beslut. Samtidigt måste systemet upprätthålla informationens *riktighet* – den får inte förvanskas då det kan leda till felaktiga och potentiellt farliga beslut. Stridsledningssystemet hanterar hemlig information varvid systemet måste upprätthålla *sekretess* – information som läcker ut kan ge motståndaren ett övertag som kan leda till omfattande konsekvenser. I det värsta scenariot är antagonisten en stormakt som kan lägga signifikanta resurser på att påverka stridsledningssystemet för att nå sina mål, varvid kraven på systemets IT-säkerhet blir mycket höga.

Oavsett hur höga krav verksamheten har på en produkts säkerhet så är det viktigt att ha tilltro till produktens kvalitet och dess pålitlighet att utföra den funktion som avsetts när produkten tas in i verksamheten. Detta gäller både för enkla produkter, såsom skruvar, och för komplexa produkter, såsom IT-system. Beroende på produktens art och komplexitet är det olika svårt att nå tillräcklig tilltro till den. En skruv går att inspektera visuellt och taktilt, exempelvis för att undersöka att den ser ut som en skruv och att den har en rimligt vikt, samtidigt som testning är relativt enkel att genomföra, exempelvis genom stickprov av draghållfasthet. Med ökande komplexitet hos produkten blir det snabbt betydligt svårare att nå samma nivå av tilltro, vilket medför att andra metoder behöver användas. Utöver inspektion och testning kan det exempelvis krävas att produkten kommer från en betrodd leverantör, att de ingående komponenterna är spårbara till betrodda tillverkare och att produkten och dess komponenter hanteras på ett spårbart sätt hela vägen från produktion till användning.

Numera är IT-system och de däri ingående delarna i form av elektronik och mjukvara mycket komplexa. En drivande trend i sammanhanget är den ökande generaliseringen av programmerbara elektronikkomponenter, såsom processorer och programmerbar logik, för att kunna täcka in en allt större marknadsandel med färre fördyrande specialiseringar av komponenterna. Detta har gjort den komplexa hårdvaran till en billig stapelvara som i sin tur tillåter att allt

komplexare mjukvara används. På mjukvarusidan finns motsvarande utveckling där färdiga komponenter såsom operativsystem, funktionsbibliotek och programvaror är den snabbaste och billigaste vägen till ett färdigt system. Detta leder dock till att även enkla systemdelar ofta realiseras med mycket komplex hårdvara och mjukvara. Komplexiteten innebär att systemdelarna blir så pass svåra att granska och testa att det med stor sannolikhet finns kvar kritiska brister eller sårbarheter som då även återfinns i det färdiga systemet.

Tilltron till systemens korrekthet är en viktig aspekt för IT-system i säkerhetskritiska tillämpningar. Samhällsviktiga verksamheter kan utsättas för angrepp och sekretessbelagd information kan läcka på grund av sårbarheter i systemen. Inga IT-system utvecklas helt i egen regi idag, då det även i enkla system ingår delar som utvecklas av andra, såsom elektronikkomponenter och biblioteksfunktioner i mjukvara. I de flesta IT-system ingår stora mängder externt utvecklade delar, exempelvis hårdvara, operativsystem och mjukvarubibliotek. För att nå tilltro till systemet som helhet krävs tillräcklig tilltro till samtliga delar i systemet – även de som utvecklats externt – och till att dessa delar samverkar på ett korrekt sätt. Som nämnts ovan är det inte praktiskt genomförbart att utföra heltäckande granskning och testning av systemet och dess delar, vilket gör det nödvändigt att bygga tilltron på andra grunder.

Under de senaste åren har det inträffat ett antal uppmärksammade händelser som påvisar hur sårbar leveranskedjan för IT-system är (se Collins, 2016; Greenberg, 2018; Khandelwal, 2018). Händelserna illustrerar även att företag och individer stundtals litar blint på produkter och tjänster som skapas av personer och organisationer som i praktiken är helt okända för dem. Denna utveckling är förståelig – det är svårt att nyttja IT utan att lita på leverantörerna och i många fall leder oönskade händelser endast till små konsekvenser – men blir problematisk när det gäller säkerhetskritiska tillämpningar. Implicit tilltro till leverantörerna och deras produkter kan leda till stora konsekvenser för exempelvis samhällsviktig verksamhet eller sekretessbelagd information, då sårbarheter i systemen kan möjliggöra oönskade händelser. Tilltro är något som lämpligen hanteras explicit, då systemens korrekthet är en central aspekt när det kommer till säkerhet. Förändringar eller felaktigheter som introduceras i systemen redan hos leverantörerna är oftast mycket svåra att upptäcka i det färdiga systemet. Hot mot cyberleveranskedjan är därför viktiga att beakta som en del i skyddet av det färdiga IT-systemet. För att kunna beakta hoten är det nödvändigt att ha god förståelse för dem samt för de risker de innebär och de motåtgärder som finns att tillgå.

Inom studien har två fallstudier genomförts för att ytterligare undersöka komplexiteten i cyberleveranskedjorna. Den första fallstudien undersöker hårdvaran i en serverdator för att illustrera såväl komplexiteten i själva hårdvaran som komplexiteten i dess leveranskedja. Den andra fallstudien undersöker dessa två

aspekter för mjukvaror med så kallad *öppen källkod* (eng. open source software). Mjukvaror med öppen källkod har valts då projekten bakom dessa är publika, vilket innebär att det finns information tillgänglig för studien.

1.1 Syfte och mål

Syftet med studien är att ge en bättre kunskap för risker som kan uppstå i cyberleveranskedjan och vad som går att göra för att motverka dessa risker. Kunskapen är avsedd att underlätta diskussioner kring cyberleveranskedjor och de associerade riskerna i samband med utveckling och förvaltning av IT-system i Försvarsmakten.

Målet med studien är att besvara följande frågor:

- Hur relevant är cyberleveranskedjeproblematik för Försvarsmakten?
- Hur komplexa är cyberleveranskedjorna för moderna IT-system?
- Vilken kunskap finns redan om cyberleveranskedjeproblematik för IT-system?
- Finns det behov av fördjupade studier kring cyberleveranskedje-problematik ur ett Försvarsmaktsperspektiv?

Studien har utförts inom ramen för Försvarsmaktens samlingsbeställning inom forskning- och teknikutveckling (FoT). Studien har genomförts i projektet IT-säkerhetsmetoder som ingår i FoT-området Operationer i cyberdomänen. Denna studie har särskilt prioriterats av Försvarsmaktens FoT-grupp. Rapporten riktar sig huvudsakligen till personer som arbetar med anskaffning, utveckling och förvaltning av Försvarsmaktens IT-system.

1.2 Definition och avgränsning

Det tycks inte finnas någon tydlig definition av cyberleveranskedjor eller någon konsensus om vad som ingår i dem. Därmed saknas en tydlig gräns för vilka risker som hör till cyberleveranskedjor och som därmed kan påverka systemens IT-säkerhet sett till denna kontext. Följande definition utgör utgångspunkt för diskussionerna i studien.

> ***Cyberleveranskedja***
>
> *Samtliga behöriga aktörer och avsedda aktiviteter som ingår i utveckling, produktion och distribution av en IT-produkt, omfattande såväl hårdvara som mjukvara.*

Definitionen innebär att cyberleveranskedjor utgörs av allt som påverkar IT-produkten fram till och med leverans. Leveransen kan ske såväl externt som

internt, vilket innebär att cyberleveranskedjan omfattar både interna och externa aktiviteter. Cyberleveranskedjor inkluderar alla steg som utförs, från produktens minsta delar till dess att den kompletta produkten driftsätts hos användarna. Dessa steg inkluderar exempelvis utveckling, integration, transport och lagerhållning. Avgränsningen till ”behöriga aktörer och avsedda aktiviteter” görs för att definitionsmässigt separera cyberleveranskedjor från de associerade IT-säkerhetsriskerna, vilka definieras nedan.

> ***IT-säkerhetsrisker från cyberleveranskedjor***
>
> *Risker som är kopplade till att IT-säkerheten i en IT-produkt påverkas negativt genom oönskade händelser i cyberleveranskedjor. Händelserna kan uppkomma såväl genom avsiktliga angrepp som genom misstag och slarv.*

Definitionen betyder att riskerna innefattar alla typer av oönskade händelser som uppstår genom påverkan på systemets delar, exempelvis genom förändringar i funktionalitet hos komponenter, men även genom andra typer av påverkan såsom förändringar i system- och användardokumentation.

Definitionen exkluderar angrepp under drift, exempelvis i form av förändringar i systemet eller dess konfiguration, då dessa sker efter att cyberleveranskedjan tagit slut i samband med leverans och driftsättning. Denna avgränsning gäller oavsett om det handlar om det ursprungliga systemet eller om systemet har uppdaterats. Även i daglig drift kan leverantörer vara involverade, exempelvis för att sköta systemadministration eller använda systemet, men dessa exkluderas av definitionen då de inte är att betrakta som *leverantörer av systemet i sig*. Däremot omfattas de personer och organisationer som är involverade i att ta fram eventuella uppdateringar och uppgraderingar. Sammantaget innebär detta att studien inte omfattar händelser som direkt orsakats av leverantörer genom deras tillgång till systemet under drift, exempelvis för dagligt driftstöd och felsökning.

Då rapporten endast behandlar cyberleveranskedjor används termerna *leveranskedja* och *cyberleveranskedja* utbytbart i texten. I de få fall där rapporten berör generella leveranskedjor, det vill säga leveranskedjor för produkter som inte ingår i IT-området, framgår detta uttryckligen i texten.

2. Bakgrund

IT-system är komplexa sammansättningar av komplexa komponenter. Hårdvaran i en kontorsdator byggs samman av tusentals delar, exempelvis mekaniska och elektroniska komponenter. De flesta elektroniska komponenter som används är enkla, såsom motstånd och kondensatorer, medan exempelvis processorer kan vara oerhört komplexa.

På datorn körs sedan flera olika mjukvaror, typiskt bestående av ett operativsystem och applikationer såsom ordbehandlare och kalkylprogram. Dessutom finns flera så kallade *inbyggda mjukvaror* (eng. firmware), vilka till stor del är dolda för användaren. Datorns inbyggda uppstartsmjukvara körs igång direkt när den slås på och ser bland annat till att operativsystemet startas. Därutöver finns inbyggd mjukvara i många andra delar av datorn och dess kringutrustning, exempelvis i nätverkskort, grafikkort, tangentbord och bildskärm. Vissa inbyggda mjukvaror lagras i respektive enhet medan andra mjukvaror laddas in av operativsystemet.

Komplexiteten i systemen innebär även att ett stort antal personer och organisationer över hela världen kan ha varit inblandade i exempelvis utveckling, produktion, distribution samt transport av systemen och deras delar. Leveranskedjorna har blivit allt mer globala och omfattar allt fler inblandade parter under senare år (Cadzow m.fl., 2015) då många system sätts samman i en räcka olika utvecklings-, produktions- och integrationssteg. Produkten som kommer ut från ett steg används som delkomponent i kedjans nästa steg.

2.1 Öppna och proprietära komponenter

Vid utveckling av IT-systemens olika delar kan de inblandade aktörerna arbeta utifrån vitt skilda förutsättningar. En viktig kategorisering av mjukvara är om den utgör det som kallas *öppen källkod* (eng. open source software) eller *proprietär mjukvara* (eng. closed source software)[1]. Öppen källkod innebär vanligen att mjukvaran med sin källkod är fritt tillgängliga och kan användas, modifieras och mångfaldigas av vem som helst under förutsättning att källkoden fortsätter att vara öppen även efter eventuella modifieringar. Proprietär mjukvara ägs i regel av ett företag eller en organisation som licensierar ut användningsrätten till användarna, ofta mot betalning. Det är sällan som användarna får

[1] *Kommersiell mjukvara* utgör inte motsatsen till öppen källkod, då den senare mycket väl kan ingå i kommersiella sammanhang. Motsatsförhållandet är istället mellan *öppen* (tillgänglig, fri, eng. open) och *proprietär* (ägd, otillgänglig, eng. closed).

tillgång till källkoden för proprietär mjukvara då den normalt sett anses vara en företagshemlighet.

Det finns en viss spridning i vad begreppen öppen källkod respektive proprietär mjukvara betyder, vilket följer av olikheter i den uppsjö av licensavtal som används. Licenser för öppen källkod skiljer sig åt genom olika grad av frihet, där vissa licenser kräver att källkod som en användare tillför till en mjukvara för egen användning måste släppas enligt samma licens. Andra licenser ger i princip användaren helt fria händer att göra som denne vill med källkoden. Det finns även en spridning i licensavtalen för proprietär mjukvara, där även källkoden kan vara tillgänglig för exempelvis vissa kunder. Detta är dock helt beroende på hur avtalet som träffats mellan leverantören och kunden ser ut. När kunden får tillgång till information som vanligtvis ses som företagshemligheter – såsom källkoden – regleras det i regel genom så kallade *sekretessavtal* (eng. non-disclosure agreements, NDA) som föreskriver en strikt kommersiell sekretess där kunden inte får sprida någon del av informationen vidare till andra parter.

De engelska begreppen open source och closed source har fått en bredare användning bortom mjukvara, där exempelvis *öppen hårdvara* (eng. open source hardware) och *öppen dokumentation* (eng. open source documentation) blivit allt populärare. Jämfört med öppen källkod är dessa områden fortfarande små, även om antalet projekt inom exempelvis området öppen hårdvara ökar i antal. Exempel inom hårdvaruområdet är *Arduino*[2], en familj med både öppen hårdvara och mjukvara som tillsammans utgör ett lättillgängligt och billigt sätt att börja med elektronik och programmering. Ytterligare exempel är de stora mängder 3D-modeller som numera finns publicerade under öppna licenser till följd av 3D-skrivarnas popularitet.

2.2 Standardprodukter

Det är önskvärt att bygga systemen med färdiga standardprodukter när så är möjligt för att undvika de höga kostnaderna förknippade med att utveckla egna IT-produkter. Standardprodukter benämns ofta med engelskans *commercial off-the-shelf (COTS)* eller de mer specifika termerna *military off-the-shelf (MOTS)* och *government off-the-shelf (GOTS)* beroende på respektive målmarknad. COTS kan inkludera såväl öppna som proprietära delar, vilka tillsammans utgör en lämpligt avgränsad och fungerande produkt. Ett exempel på en COTS-produkt med blandade licensmodeller är företaget Netgates *Security Gateway*

[2] http://arduino.cc

Appliances[3]. Det är en produktserie bestående av proprietär hårdvara som designats för att köra *pfSense*, en brandväggsmjukvara som är öppen källkod och som även är förinstallerad i produkterna.

Många IT-system är specialutvecklade eller anpassade specifikt för de behov som systemen ska möta. Specialutvecklade system är vanligt inom exempelvis Försvarsmakten, där verksamheten i många avseenden skiljer sig från verksamheter inom den i civila sfären. Grunden i de specialutvecklade systemen är i regel vanliga standardkomponenter, såsom färdiga datorer, operativsystem och COTS-mjukvara. Standardkomponenterna sätts samman till ett grundsystem som sedan kompletteras eller anpassas med egenutvecklade komponenter för att skapa det färdiga systemet.

2.3 Komplexitet hos IT-system

Processorer, minnen och många andra komponenter i dagens datorer är så kallade *integrerade kretsar*, det vill säga komponenter som i sig innehåller en elektronisk konstruktion bestående av delfunktioner som kopplats samman internt. En fundamental byggsten i de integrerade kretsarna är *transistorn*, vars funktion förenklat kan ses som en elektriskt styrbar strömbrytare. Transistorn är den byggsten som möjliggör alla digitala funktioner som ger datorerna deras funktion. Den integrerade kretsen uppfanns i slutet av 1950-talet men det dröjde till 1971 innan lanseringen av Intel 4004, den första publikt tillgängliga[4] integrerade processorn, bestående av cirka 2250 transistorer.

1965 skrev Gordon E. Moore (2006a), en av grundarna till Intel, en artikel där han förutspådde att det maximala antalet transistorer på en integrerad krets ungefär skulle fördubblas varje år samtidigt som kostnaden för kretsarna skulle sjunka. Tio år senare reviderade Moore detta till att tempot skulle hålla i sig till ungefär 1990 för att sedan sjunka till en fördubbling ungefär var annat år (Moore, 2006b). *Moores lag*, som detta har kommit att kallas, höll ganska bra ända fram till 2010-talet, då utvecklingstakten börjat sjunka till följd av svårhanterade fysikaliska fenomen i tillverkningsprocessen. Effekten av utvecklingen är att det idag finns otroligt komplexa processorer, exempelvis Qualcomm Centriq 2400 med 18 miljarder transistorer i en och samma krets (Qualcomm, 2017).

[3] https://www.pfsense.org/products/

[4] På 1990-talet släpptes tidigare sekretessbelagd information om en processor som byggdes specifikt för stridsflygplanet F-14 Tomcat. Denna processor började produceras 1970, det vill säga året innan lanseringen av Intel 4004.

Sedan den integrerade processorns introduktion för snart 50 år sedan har antalet transistorer per krets därmed ökat med ungefär 10 miljoner gånger.

Kostnaden för att utveckla en komplex integrerad krets är naturligtvis hög. För standardkomponenter såsom processorer och minnen kompenseras detta genom stora volymer med producerade kretsar som var och en endast behöver bekosta en liten del av utvecklingskostnaden. Som indikation på volymerna såldes mer än 250 miljoner persondatorer globalt under 2018. Till det kommer alla så kallade inbyggda datorsystem, det vill säga mer eller mindre komplexa datorer som ingår som del i ett större system eller en maskin. En stor leverantör inom inbyggda system är ARM, vars processorkonstruktioner integrerats i många andra leverantörers kretsar. Under åren 2013–2017 levererades ungefär 50 miljarder ARM-processorer och det totala antalet överstiger 100 miljarder sedan introduktionen 1991 (Hughes, 2017).

Mjukvaran har gått en liknande väg, där den alltmer kraftfulla hårdvaran har gett möjlighet att implementera allt fler funktioner och finesser i mjukvaran vilket även lett till allt mer komplex och omfattande mjukvara. Förutom den synliga mjukvaran i form av operativsystem och applikationsprogram innehåller datorer även så kallad inbyggd mjukvara. Exempelvis innehåller en kontorsdator med Microsoft Windows 7 och Office 2013 samt inbyggd mjukvara en total mängd mjukvara som motsvarar i storleksordningen 100 miljoner rader källkod (McCandless, Doughty-White, & Quick, 2015).

Numera finns enorma mängder tillgänglig mjukvara i form av källkodsbibliotek och färdig mjukvara. Detta är delvis ett resultat av den kraftiga framväxten av öppen källkod, men bygger även på att de nödvändiga förutsättningarna är tillräckligt lättillgängliga. Sådana förutsättningar är exempelvis tillgång till datorer, utvecklingsverktyg och kunskap. Tillgången till källkodsbibliotek och färdig mjukvara innebär att utvecklare inte behöver implementera vanligt förekommande funktionalitet då denna redan finns tillgänglig. Därmed går det snabbare att nå fram till fungerande system, vilket i sin tur medför lägre utvecklingskostnad och kortare utvecklingscykler. En nackdel med denna typ av återanvändning är att de färdiga funktionsbiblioteken ofta innehåller betydligt fler funktioner än vad som behövs i respektive system, vilket innebär att systemen tillförs komplexitet som inte bidrar till funktionaliteten.

Utvecklingen mot att högre komplexitet ger lägre kostnad är närmast en anomali sett till traditionella industrier, där en mer komplex produkt tenderar att kosta mer att utveckla och producera än en enklare produkt. I IT-sammanhäng gäller inte längre kopplingen mellan komplexitet och kostnad, då framstegen enligt Moores lag har lett till generell hårdvara med breda användningsområden och stora tillverkningsserier, vilket i sin tur ger låga priser trots den höga komplexiteten. Mjukvaran är ännu mer speciell då marginalkostnaden i princip är noll för

att mångfaldiga mjukvara samtidigt som tillgången på fri mjukvara är stor. Det ska dock inte tolkas som att fri mjukvara eller öppen källkod är dålig, då det inte går att dra den typen av slutsatser utifrån licensmodellen. Det är huvudsakligen andra faktorer som påverkar kvaliteten, exempelvis mognadsgrad i utvecklingsarbetet.

En viktig insikt när det gäller komplexiteten är att utvecklare utgår från en generell lösning – datorn – som i princip skulle kunna lösa vilket problem som helst. Genom mjukvara appliceras sedan begränsningar för att få datorn att lösa det specifika problem som utvecklarna i fråga arbetar med. De har därmed börjat med den generella datorn – som kan anta vad som i praktiken är oändligt många tillstånd[5] – och begränsat den till att endast vara i de tillstånd som anses vara produktiva och säkra. Denna form av *simulerad enkelhet* döljer komplexiteten i systemet, men döljer även risker då små misstag eller förändringar i tillståndet potentiellt kan leda till stora effekter på systemets beteende (Dullien, 2018).

Med komplexiteten i en enskild dator som utgångspunkt går det att vidga resonemanget till IT-system bestående av sammankopplade datorer eller sammankopplade IT-system. Även om datorerna till stor del består av liknande hårdvara och mjukvara så finns det i många fall variationer mellan dem. Till exempel nyttjas liknande datorkomponenter från olika leverantörer och olika versioner av mjukvara. I nätverken som knyter samman datorerna finns utrustning såsom switchar och routrar, som även de innehåller komplex hårdvara och mjukvara. Interaktion mellan olika datorsystem och kommunikation över otillförlitliga kanaler (såsom internet) adderar komplexitet i den totala systembilden. Dessutom kan systemen interagera med användare av olika bakgrund, med olika kompetens, med olika drivkrafter, med olika värderingar och som ibland även styrs av olika regelverk. Detta ger ytterligare dimensioner av komplexitet i den totala systembilden.

Alla delar i IT-system kan innehålla brister, både i säkerhetsmekanismer och i andra funktioner. Genom systemens komplexitet kan bristerna innebära sårbarheter som är mycket svåra att hitta. Antalet sårbarheter i systemen kan sannolikt minska med bra utvecklingsmetoder samt väl genomförd validering och verifiering av systemen. Det till trots kan sårbarheter dyka upp i mjukvaran då helt säker kod är närmast omöjlig att uppnå.

[5] I teorin är antalet tillstånd begränsat, men ur alla praktiska hänsyn är antalet att betrakta som oändligt. En megabyte minne motsvarar $2^{8 \cdot 1024 \cdot 1024}$ tillstånd – ett tal med drygt 2500000 siffror. Moderna datorer har miljontals gånger mer lagring, vilket leder till att den kan befinna sig i enormt många olika tillstånd.

3. Cyberleveranskedjor

Enligt definitionen i avsnitt 1.2 består en cyberleveranskedja av behöriga aktörer och avsedda aktiviteter som ingår i utveckling, produktion och distribution av en IT-produkt, innefattande både hårdvara och mjukvara. Detta inkluderar alltså personal inom systemarkitektur, utveckling, testning, produktion, logistik, lagerhållning, transport, integration, installation och många andra områden som utför en uppsjö olika arbetsuppgifter.

Eftersom IT-produkter i regel består av många delkomponenter (i såväl hårdvara som mjukvara) från olika leverantörer på en global marknad, följer att de personer som kommer i kontakt med produkterna arbetar på ett stort antal olika företag och organisationer i många länder. Detta kapitel tar upp två aspekter på leveranskedjornas uppbyggnad – artefakterna som produceras och hur dessa kan spåras genom leveranskedjegrafer.

3.1 Fysiska och virtuella artefakter

Varje steg i en cyberleveranskedja skapar *artefakter* som ska skickas vidare till nästa steg eller till en användare. En viktig distinktion är den mellan fysiska och virtuella artefakter, då dessa har olika egenskaper såväl i en leveranskedja som hos användaren. En *fysisk artefakt* innebär en konkret och påtaglig fysisk sak, typiskt en hårdvara eller en kombination av hårdvara med installerad mjukvara. En *virtuell artefakt* är icke-fysisk och hanteras vanligtvis i digitala format på IT-system. Typexempel på virtuella artefakter är källkod, kompilerad mjukvara, digital dokumentation och installationsavbildningar. Virtuella artefakter kan distribueras på fysiska media såsom CD-skivor eller USB-minnen, men kan lika gärna distribueras digitalt, exempelvis via webbplatser, molntjänster eller epost.

En stor skillnad mellan fysiska och virtuella artefakter är att de senare i regel är oändligt duplicerbara. Det går alltså att skapa kopior av de virtuella artefakterna med ingen eller försumbar marginalkostnad per kopia. Ett vanligt exempel är nedladdningsbar mjukvara där varje nedladdning i princip utgör en kopia. En fysisk artefakt kräver material (råvaror eller komponenter) och en tillverkningsmiljö för att kunna skapa ytterligare ett exemplar, vilket leder till högre marginalkostnad.

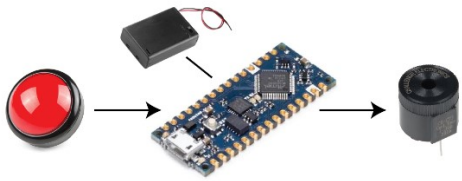
Genom sin oändliga duplicerbarhet kan virtuella artefakter genomgå en kvalitetskontroll och säkerhetsgranskning för att sedan mångfaldigas och användas där det behövs. Detta kan exempelvis vara en mjukvara eller mjukvaruuppdatering som granskas och godkänns en gång. Därefter kan installation genomföras utan att ytterligare kontroller är nödvändiga. Detta kan kontrasteras med hårdvara, exempelvis datorer, där varje exemplar i princip

måste kontrolleras med avseende på kvalitet och säkerhet för att säkerställa att eventuella problem upptäcks.

3.2 Leveranskedjegraf

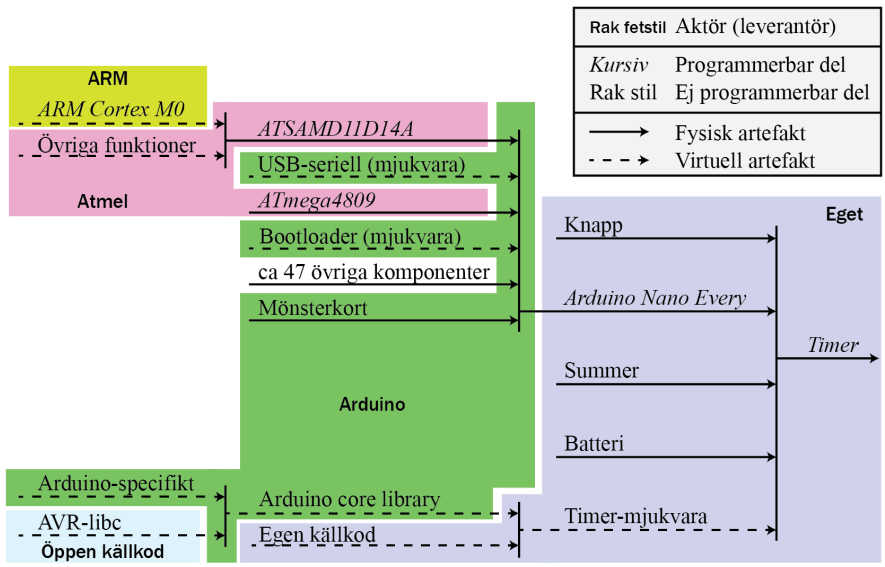
Om alla artefakter spåras genom alla led i leveranskedjan för en produkt bildas en trädliknande riktad graf, där produkten utgör slutnoden och förgreningspunkterna motsvarar de förädlingssteg där produktens delar utvecklas eller sätts samman. Löven motsvarar de ursprungliga artefakterna, de minsta komponenterna i produkten. En sådan *leveranskedjegraf* visar relationer mellan systemets delar och hur dessa relaterar till exempelvis de producenter, distributörer och transportörer som varit involverade i utveckling och hantering av produkten. Grafen kan fyllas på med detaljer ner till enskilda kodrader och utvecklare samt de grundmaterial som används för att producera hårdvara. Sett ur IT-säkerhetsperspektiv är denna nivå rimligtvis helt onödig och dessutom otroligt komplex för de allra flesta produkter. En sannolikt rimligare nivå är att visa vilka organisationer som varit inblandade och vad de gjort i respektive förädlingssteg med vilken del eller komponent i produkten, samtidigt som spårningen begränsas till ett relevant djup för olika typer av artefakter.

Även med en relativt begränsad informationsmängd och en enkel produkt blir leveranskedjegrafen snabbt komplext. För att illustrera den komplexitet som nås utgår följande exempel från en enkel produkt – en timer som kan byggas av en person på hobbynivå, utan omfattande förkunskaper eller dyra verktyg. Timern har en knapp som startar nedräkningen av en förutbestämd, fast tidsperiod och en summer som ljuder när tiden är ute. För att förenkla elektroniken utgår exemplet från en färdig modul, *Arduino Nano Every*, med en enkel mikroprocessor och några kringkomponenter. Arduino-modulen är öppen hårdvara, vilket innebär att kretsschema, komponentlista och annan information om modulen är fritt tillgänglig.



Figur 1. Schematisk skiss över timerns hårdvara, bestående av en knapp, en Arduino-modul och en summer med spänningsmatning från ett batteri.[6]

Figur 1 visar en schematisk skiss över hårdvaran i timern. Notera att hårdvaran i princip består av tre delar (plus spänningsförsörjning i form av ett batteri), där den överlägset mest komplexa delen är Arduino-modulen. Det är även den modulen som innehåller all mjukvara i timern.



Figur 2. Förenklad leveranskedjegraf för timern. Färgerna grupperar respektive leverantörs artefakter.

[6] Bilderna på knappen, Arduino-modulen, summern och batteripacket är hämtade från *Sparkfun.com*. Bilderna är licensierade under CC-BY 2.0, se https://creativecommons.org/licenses/by/2.0/.

Figur 2 visar en mycket förenklad leveranskedjegraf för timern, där endast de artefakter som är distinkta och tydligt identifierbara tas upp. Horisontella linjer motsvarar delkomponenter i timern och hur de relaterar till olika förädlingssteg. Flödet i grafen går från vänster till höger där förädlingsstegen markeras av vertikala linjer. Notera att trädet har förenklats avsevärt genom att alla icke-programmerbara elektronikkomponenter på Arduino-modulen har slagits samman som ”ca 47 övriga komponenter”. Mjukvaran är också mycket enkel i produkten, då inga tredjepartsbibliotek används och den egna källkoden består av cirka 25 rader.

Grafen i figur 2 visar dock bara de relationer som är direkt kopplade till timerns tekniska uppbyggnad. Alla interaktioner mellan leverantörer och de olika artefakterna har utelämnats, både för överskådlighetens skull och för att dessa i många fall är svåra att ta reda på. Exempelvis produceras modulen gissningsvis inte av företaget Arduino AG som marknadsför den utan av någon under-leverantör. Leverantörsinteraktioner som inte innebär någon förädling är också utelämnade från figuren. Detta inkluderar exempelvis distribution och transport.

Exemplet ovan visar att det krävs mycket enkla produkter för att leveranskedje-grafen ska bli överskådlig. IT-system är generellt sett mycket mer komplexa, varför grafen snabbt blir väldigt komplex och svåröverskådlig. Det är många olika aktörer inblandade i utveckling, produktion, integration, distribution, transport och underhåll av IT-system. I praktiken är det svårt att nå mer än några få nivåer ner i grafen innan den nödvändiga informationen blir svåråtkomlig eller komplexiteten blir för hög.

Som nämnts ovan kan spårningen i leveranskedjegrafen i teorin leda ända till råmaterial, men det djupet är knappast vare sig rimligt eller nödvändigt för att kunna bedöma leveranskedjan ur ett IT-säkerhetsperspektiv. Dessutom är det sällan att alla delar är lika viktiga för systemets IT-säkerhetsegenskaper. Genom att fokusera på de artefakter som är viktigast för IT-säkerheten går det att minimera grafen, men det kommer sannolikt fortfarande vara komplext att skapa en graf för verkliga produkter och system.

4. Komplexitet i cyberleveranskedjor

Detta kapitel presenterar resultaten från två fallstudier som belyser komplexiteten i dagens IT-system och deras leveranskedjor. Det första fallet fokuserade på hårdvara i form av en serverdator och undersökte såväl serverns modulära uppbyggnad som de unika dator- och elektronikkomponenter som ingick i servern. Det andra fallet fokuserade på ett antal mjukvaror och undersökte respektive mjukvaras komplexitet och hur utvecklingen skett.

4.1 Fall 1: Hårdvara

Hårdvaran som undersökts var en server från HP av modellen *Proliant SL2x170z G6* som producerades cirka år 2009. Då undersökningen varit förstörande användes en äldre, utrangerad server som utgångspunkt för att inte behöva förstöra användbar utrustning. Resultaten från studien anses representativa trots att servern är relativt gammal, då datorhårdvara fortfarande byggs enligt samma principer och med liknande uppbyggnad som gällde när servern i fråga producerades.

Figur 3. Serverhårdvaran som undersökts.

Figur 3 visar serverhårdvaran som undersökts. Det är en rackmonteringsbar multiserver med två fack som vardera innehöll två servrar. Serverhårdvaran bestod alltså av fyra separata servrar med ett gemensamt nätaggregat. De fyra interna servrarna var identiska ur ett hårdvaruperspektiv.

Då syftet med fallstudien var att undersöka komplexiteten i hårdvaran och inventera vilka leverantörer som varit inblandade i produkten så utgick undersökningen från att såväl datorkomponenter som elektronikkomponenter var äkta. Därmed utgick undersökningen även från att alla beteckningar som återfunnits på respektive komponent var korrekta. Då det är unika komponentmodeller som påverkar leveranskedjans komplexitet, snarare än det totala antalet komponenter, fokuserar studien på komponenter med unika modellbeteckningar.

Första analysnivån i fallstudien undersökte *datorkomponenterna* i servern, det vill säga moderkort, processorer, minnen, hårddiskar, insticksкort och adaptrar. Nätaggregatet exkluderades ur analysen då det var av mindre intresse sett ur ett IT-säkerhetsperspektiv.

Den andra analysnivån undersökte *elektronikkomponenterna* som var fastlödda på servens moderkort. Denna analysnivå avgränsades till att endast undersöka elektronikkomponenterna på moderkortet, då det ansågs vara tillräckligt för att belysa de frågor som fallstudien undersökte. Om undersökningen hade syftat till att hitta manipulerad hårdvara i servern hade elektronikkomponenterna i samtliga datorkomponenter varit av intresse. En ytterligare avgränsning var att ingen djupare analys genomfördes av de identifieringstexter som återfanns på komponenterna, utöver de som utgjorde någon form av modellbeteckning.

Merparten av elektronikkomponenterna innehöll inga programmerbara funktioner utan var statiska i sin funktion. Exempel på komponenter med statisk funktion är enklare integrerade kretsar och så kallade diskreta komponenter, såsom motstånd och kondensatorer. Andra elektronikkomponenter var mer komplexa och inkluderade exempelvis programmerbara funktioner eller minne. Följande kategorisering används för elektronikkomponenter:

1. programexekverande[7] med permanent minne
2. programexekverande utan permanent minne
3. permanent minne
4. flyktigt minne
5. övriga.

De elektronikkomponenter som är mest intressanta ur IT-säkerhetshänseende när det gäller manipulation är de som innehåller permanenta minnen, det vill säga kategori 1 och 3. Innehållet i dessa minnen kan i många fall ändras utan att någon åverkan behöver göras på kretskorten. Andra typer av manipulation kan påverka komponenter ur de andra kategorierna, exempelvis genom utbyte mot manipulerade komponenter vid produktion. Dessa angrepp är dock i de flesta fall mer avancerade jämfört med att programmera om ett minne.

Denna analysnivå avgränsades till att undersöka elektronikkomponenterna på moderkortet. Samtliga integrerade kretsar på moderkortet undersöktes för att bestämma kretsens funktion och därmed vilken kategori de tillhör. I servern förekom det processorer, programmerbara logikkretsar, permanenta minnen, flyktiga minnen, strömförsörjningskretsar, nätverkskretsar samt enklare analoga

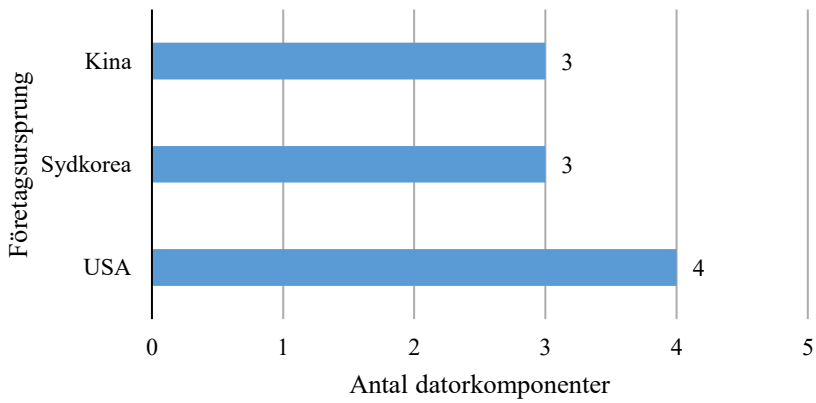
[7] *Programexekverande* innebär att komponenten innehåller någon form av funktioner som exekverar mjukvara, något som till exempel finns i processorer och programmerbar logik.

och digitala kretsar. Utöver dessa fanns även ett stort antal diskreta komponenter, såsom motstånd, kondensatorer, induktorer och transistorer.

En observation som gjordes vid demonteringen av servern är att flera datorkomponenter var märkta med en säkerhetsetikett från HP, där det fanns ett identitetsnummer som gör det möjligt för mottagaren av hårdvaran att verifiera datorkomponentens autenticitet[8]. Säkerhetsetiketterna inkluderar ett holografiskt element och lämnar delar av trycket efter sig om de tas bort, vilket ger en viss grad av skydd mot förfalskning av etiketterna. De datorkomponenter som hade denna märkning var RAM-minnen, processorer och hårddiskar. En möjlig anledning till att det är just dessa datorkomponenter som har säkerhetsetiketter är att de är lätt utbytbara och relativt dyra, vilket innebär att de är attraktiva att förfalska.

4.1.1 Datorkomponenternas ursprung

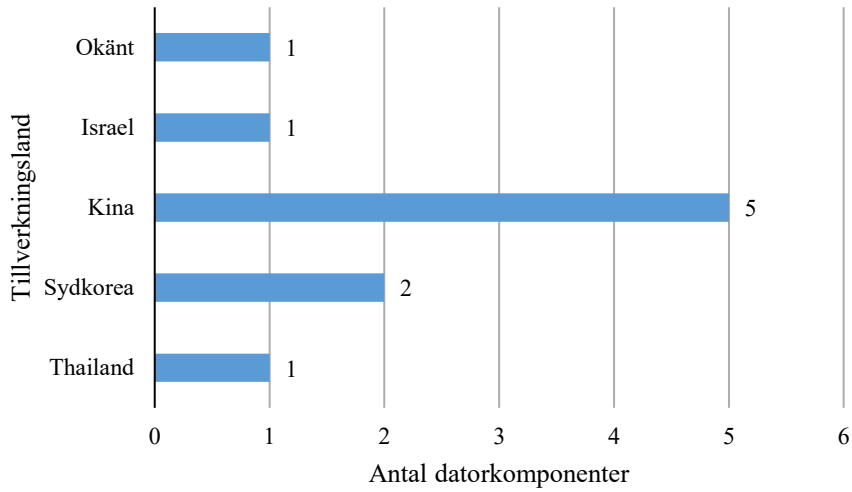
Servern består av tio datorkomponenter (med unik modellbeteckning) som tillverkats av sex företag. Figur 4 visar ursprunget för datorkomponenterna, baserat på var företaget som låtit tillverka respektive komponent kommer från. Samtliga komponenter har sitt ursprung i antingen USA, Sydkorea eller Kina.



Figur 4. Företagsursprung för datorkomponenter i servern.

[8] https://www.hpe.com/us/en/validate.html

Många datorföretag är internationella med fabriker i flera länder. Detta innebär att tillverkning av datorkomponenter många gånger sker i andra länder än där producenten har sitt säte. När det gäller vilka länder som datorkomponenterna tillverkas i ser andelarna därför markant annorlunda ut. Figur 5 visar fördelningen mellan tillverkningsländer, där nära hälften av datorkomponenterna är tillverkade i Kina. Att just Kina sticker ut är förväntat då landet är en stor producent av elektroniska produkter och många internationella företag har sin tillverkning där. Tillverkningen av övriga datorkomponenter är relativt jämnt fördelad mellan Israel, Sydkorea och Thailand. Vissa datorkomponenter har okänt tillverkningsland.



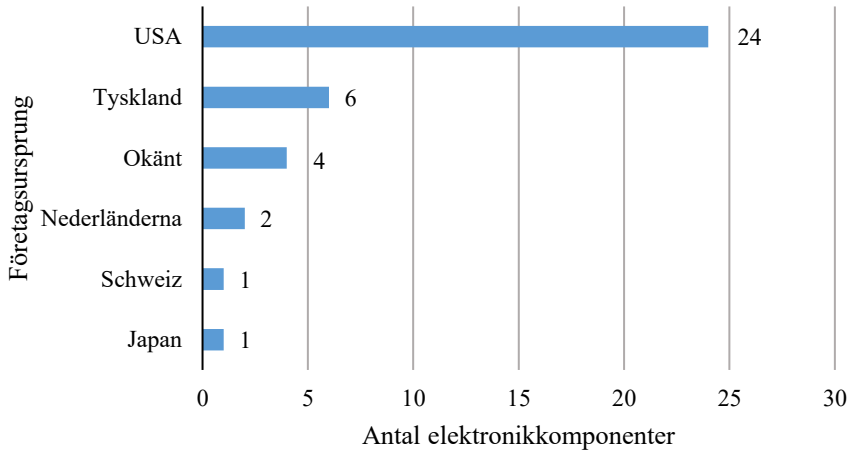
Figur 5. Tillverkningsländer för datorkomponenter i servern.

4.1.2 Elektronikkomponenternas ursprung

I detta avsnitt analyseras de *elektronikkomponenter* som är fastlödda på moderkortet. Analysen avgränsades till att endast omfatta aktiva komponenter, vilket innebär att passiva komponenter såsom resistorer, kondensatorer och induktorer inte har undersökts. Resultatet från inventeringen visar att det är många företag som var involverade i att ta fram de komponenter som behövdes för att tillverka moderkortet. För de 38 komponenter (med unik modellbeteckning) som inventerades var åtminstone 16 olika företag inblandade. Tillverkare gick dock inte att identifiera för fyra av komponenterna, vilket innebär att ytterligare företag kan ha varit inblandade.

Tillverkarna identifierades genom sökningar på internet efter beteckningar och företagslogotyper. Eftersom hårdvaran som studerades var från omkring 2009 har flera företag blivit uppköpta eller bytt namn. I de fall företaget blivit uppköpt har det uppköpande bolaget noterats.

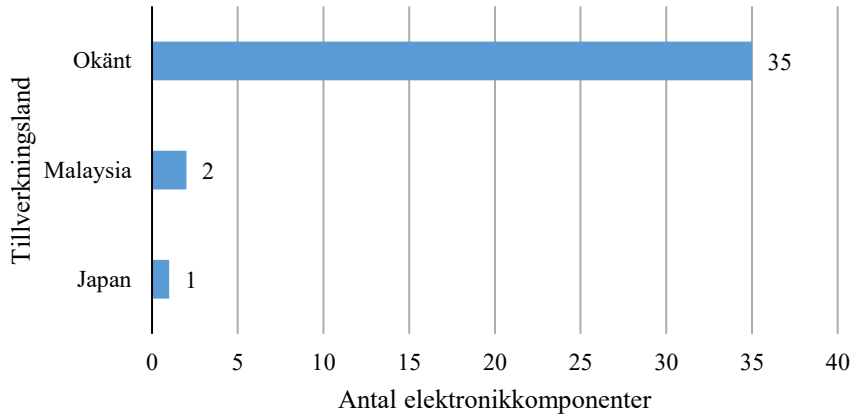
Figur 6 visar vilka länder som tillverkarna av elektronikkomponenterna har sitt säte. Företagen kommer huvudsakligen från USA följt av Tyskland.



Figur 6. Företagens ursprung för elektronikkomponenterna på serverns moderkort.

Elektronikkomponenternas producenter är ofta internationella. Detta innebär att det är vanligt att elektronikkomponenterna inte tillverkas i det land där företaget har sitt säte. Figur 7 visar resultatet från inventeringen och det framgår att tillverkningsland inte har kunnat identifieras för majoriteten av komponenterna. Tillverkningslandet identifierades endast för tre komponenter, där två hade tillverkats i Malaysia och en i Japan.

Anledningen till att tillverkningsländer inte gått att identifiera för majoriteten av komponenterna är att landet inte framgått tydligt på respektive komponent. Komponenterna kan dock ha identifierande texter som indikerar tillverkningsland, men dessa beteckningar är inte standardiserade utan skiljer sig åt mellan olika företag vilket gör det svårt att uttyda vad de betyder. Dessa identifieringstexter har dock inte studerats då de ligger utanför studiens avgränsningar.



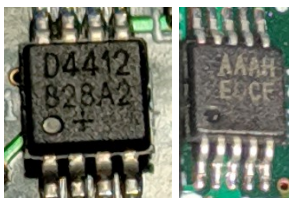
Figur 7. Tillverkningsland för elektronikkomponenter på serverns moderkort.

Till största del innehåller elektronikkomponenterna inte några programmerbara funktioner utan är statiska i sin funktion. Ett mindre antal är mer komplexa och tillhör kategorin minnen, processorer och programmerbara logikkretsar. Tabell 1 visar en sammanställning över antalet aktiva elektronikkomponenter med unik modellbeteckning som återfanns på moderkortet.

Funktionen hos de två elektronikkomponenterna som visas i figur 8 kunde inte fastställas. Som syns i figuren ryms inte mycket information på sådana komponenter, vilket kan göra det mycket svårt att utröna vilka företag som låtit tillverka dem och var de tillverkats.

Tabell 1. Antal aktiva elektronikkomponenter med unik modellbeteckning.

Typ av komponent	Kategori	Unika modeller
Programmerbar logikkrets	1	1 st
Processor utan permanent minne	2	1 st
Permanent minne	3	4 st
Flyktigt minne	4	1 st
Nätverkskrets	5	1 st
Övriga, till exempel spänningsförsörjning	5	28 st
Komponent med oidentifierad funktion	Okänd	2 st
Totalt antal unika aktiva komponenter		*38 st*



Figur 8. Komponenter som inte gått att identifiera.

4.1.3 Reflektioner

Att undersöka vilka leverantörer som bidragit till hårdvaran visade sig vara svårt och tidskrävande. För den första analysnivån, datorkomponenterna, var det relativt okomplicerat. Redan på den andra analysnivån, elektronikkomponenterna, blev det betydligt svårare.

En bidragande faktor kan vara att komponenterna är mycket små och att det inte finns mycket plats för identifierande information. Tryck på komponenterna kan användas för att ange tillverkare, modellbeteckning och tillverkningsland, men detta tar relativt mycket plats. I denna studie undersöktes inte betydelsen hos de tillverkarspecifika identifieringstexterna på komponenterna, för att exempelvis hitta information om tillverkningsland, då detta ligger utanför avgränsningarna för studien. Dessa beteckningar är specifika för respektive tillverkare vilket kan medföra en betydande arbetsinsats för att hitta information om hur de ska uttydas. Dessutom är det möjligt att information som krävs för att tolka texterna inte är tillgänglig.

Även med utgångspunkten att komponenter och beteckningar är äkta har det varit svårt att identifiera alla komponenter i servern. Därför är en slutsats att det troligen är mycket svårt att dessutom upptäcka om komponenter blivit utbytta

eller manipulerade. En sådan granskning torde kräva omfattande kunskap om vilka komponenter som borde finnas i servern och exakt hur dessa ska se ut och fungera. Dessutom är det sannolikt att en granskare måste besitta expertkunskap inom elektronik för att kunna genomföra en meningsfull granskning.

4.2 Fall 2: Mjukvara

Den andra fallstudien fokuserade på att hitta information om komplexitet, beroenden och leveransförhållanden för mjukvara. Huvudsakligen undersöktes ett antal mjukvaror med öppen källkod, då det fanns lättillgänglig och öppen information om dessa på internet.

För att analysen ska få en god spridning valdes mjukvaror med olika användningsområden och en stor användarbas. Avsikten med detta var att täcka in olika funktionalitet och ge verklighetsbaserade exempel på komplexitet. De mjukvaror som ingår i fallstudien är *Notepad++*, *OpenSSH*, *OpenVPN*, *Firefox*, *Chromium och Libreoffice*. Tabell 2 listar den grundläggande informationen för respektive mjukvara.

Då ämnet är omfattande avgränsades analysen till mjukvaruberoenden med avseende på *organisation*, *kodstorlek*, *implementationsspråk* och *utvecklare*. Informationen som analysen baserats på är öppet tillgänglig och hämtades från webbplatsen Open Hub[9] under oktober 2019.

4.2.1 Organisation

Som tabell 2 visar finns det olika organisationsformer bakom utveckling av mjukvara. Förutom mjukvara som utvecklas av privatpersoner organiseras projekten även genom *företag* och *stiftelser*. Hur företag och stiftelser sköts ser olika ut i olika länder. Det kan vara värt att nämna att mjukvaror med öppen källkod inledningsvis ofta utvecklas av privatpersoner, ibland samarbetande i vad som närmast kan liknas vid föreningsform. I vissa fall tas sedan källkoden från dessa projekt och kommersialiseras, antingen av utvecklarna själva eller av andra aktörer. Ett exempel på det senare är att Apples Mac Os X inkluderar källkod från FreeBSD[10]. Att återanvända mjukvara och källkod från andra personer och företag är vanligt förekommande inom mjukvaruutveckling

[9] https://www.openhub.net/

[10] https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/BSD/BSD.html

eftersom det ofta är enklare och billigare att använda befintlig källkod än att utveckla på nytt så länge källkoden håller tillräcklig kvalitet.

Tabell 2. Översikt över mjukvarorna.

Mjukvara	Beskrivning	Utgivare och organisationsform	Primär licenstyp	Land
Notepad++	Textredigerare	Privatperson	Öppen källkod	Frankrike
OpenSSH	Kryptering och nätverkstjänster	The OpenBSD Foundation (icke vinstdrivande företag)	Öppen källkod	Kanada
OpenVPN	VPN-tunnel	OpenVPN project, OpenVPN inc. (företag)	Öppen källkod	USA
Firefox	Webbläsare	Mozilla Corp. (icke vinstdrivande företag)	Öppen källkod	USA
Chromium	Webbläsare	Google (företag)	Öppen källkod	USA
Libreoffice	Kontorsprogram	The Document Foundation (stiftelse)	Öppen källkod	Tyskland

De undersökta mjukvarorna har utvecklats i olika länder. Medan *land*-kolumnen i tabell 2 visar var organisationen är registrerad så kan utvecklarna för mjukvaran i fråga finnas i flera länder världen över. Större mjukvaruprojekt med öppen källkod, särskilt operativsystem, utvecklas oftast kollektivt och samarbetet sker via distribuerade versionshanteringsprogram såsom Git[11]. Själva utvecklingen sker såväl i företagsmiljö som i andra miljöer, exempelvis i hemmet, medan samarbetet sker över internet.

4.2.2 Programmets storlek och språk

Tabell 3 visar antal källkodsrader för respektive mjukvara och vilka språk som dessa implementerats i så som det såg ut vid tidpunkten för datainsamling i fallstudien[12]. Spridningen i storlek är relativt stor, från drygt 100 000 rader till

[11] https://git-scm.com/

[12] Open Hub indexerar projekten vid olika tidpunkter, vilket innebär att det är olika datum som gäller för respektive mjukvara. Vid tidpunkten för fallstudiens datainsamling var de senaste indexeringarna som följer: Notepad++ 2019-11-28, OpenSSH 2019-11-28, OpenVPN 2019-04-16, Firefox 2019-09-04, Chromium 2019-08-22 och LibreOffice 2019-10-29.

knappt 26 miljoner rader, vilket även reflekterar den stora skillnaden i funktion hos respektive mjukvara. Den minsta kodbasen bland de undersökta mjukvarorna tillhör OpenSSH, ett bibliotek med kryptografiska funktioner och nätverksfunktioner. Den största kodbasen tillhör Chromium som är en webbläsare.

Tabell 3. Mjukvarornas kodstorlek och språk.

Mjukvara	**Kodstorlek**[13]	**Språk**[14]
Notepad++	551 235	54% C++ 35% XML 11% Annat
OpenSSH	126 709	86% C 9% Shell Script 5% Annat
OpenVPN	307 089	92% C 8% Annat
Firefox	20 548 088	28% C++ 26% JavaScript 16% Annat 12% HTML 12% C 6% Rust
Chromium	25 670 029	49% C++ 16% Annat 10% HTML 10% C 10% JavaScript 5% XML
Libreoffice	9 517 397	65% C++ 26% XML 9% Annat

En majoritet av källkoden är skriven i C och C++ för de mjukvaror där implementationsspråken är kända. Dessa språk är välanvända och etablerade i mjukvaruindustrin.

[13] Kodrader utan kommentarer eller tomma rader

[14] Endast språk som utgör minst 5% av programmet, språk under 5% räknas som “Annat”.

4.2.3 Beroenden

Utöver mjukvarorna i sig har fallstudien även undersökt vilka beroenden till annan mjukvara som finns. Denna del av undersökningen syftade till att illustrera omfattningen i beroenden för mindre och större mjukvaror. Därför har undersökningen begränsats till två mjukvaror, OpenVPN och Firefox, som anses representativa för de två storleksskikten. Detta avsnitt visar beroendegrafer (eng. dependency trees) för mjukvarorna så som de ser ut under Linux-distributionen Debian version 10. Programmet *debtree* nyttjades för att ta fram beroende-graferna. Detta program återfinns i flera Linux-distributioner utöver Debian.

Följande kommando användes:

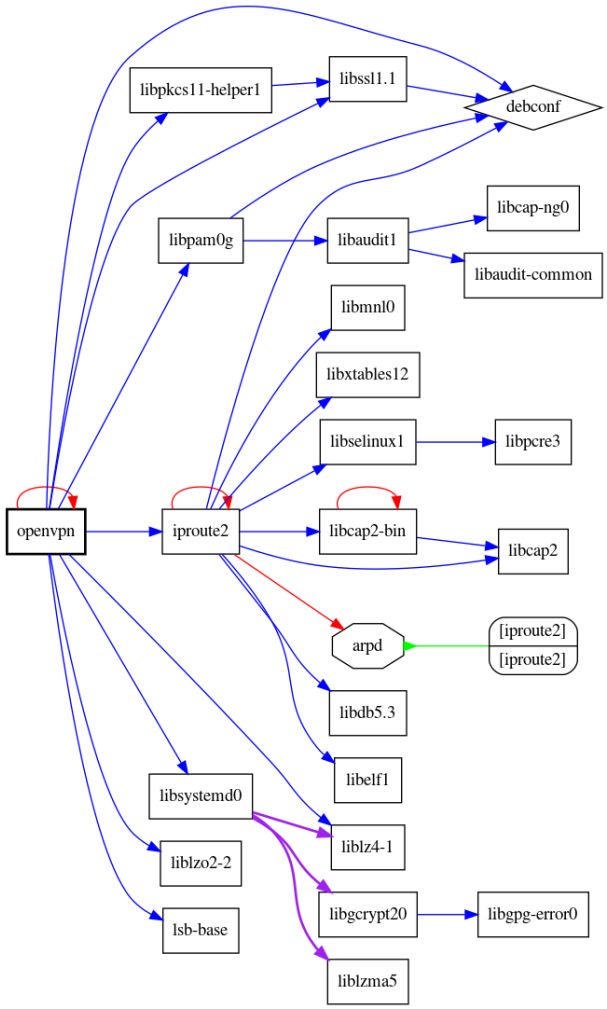
```
app=openvpn && debtree --no-versions --no-recommends
--no-alternatives --no-provides $app > $app.dot && dot
-Tpng $app.dot -o $app.png
```

För att förenkla diagrammen inkluderas endast obligatoriska beroenden i trädet. Beroenden som inte är nödvändiga, såsom rekommendationer eller alternativ i Linux-distributionen, har därvid exkluderats.

Figur 9 visar beroendeträdet för OpenVPN som det produceras av debtree. De viktigaste elementen i beroendegrafen är dessa:

- rektangel – mjukvarupaket
- oktagon – virtuellt mjukvarupaket
- rundad rektangel – valbara mjukvarupaket som implementerar ett virtuellt paket
- blå eller lila pil – beroende
- röd pil – konflikt
- grön pil – valbar implementation.

Två beroenden, *libc* och *zlib*, har utelämnats från diagrammet men är en del av beroendeträdet. Dessa utelämnades för att göra diagrammet mer överskådlig, då nästan alla program använder sig av dessa två mjukvaror.

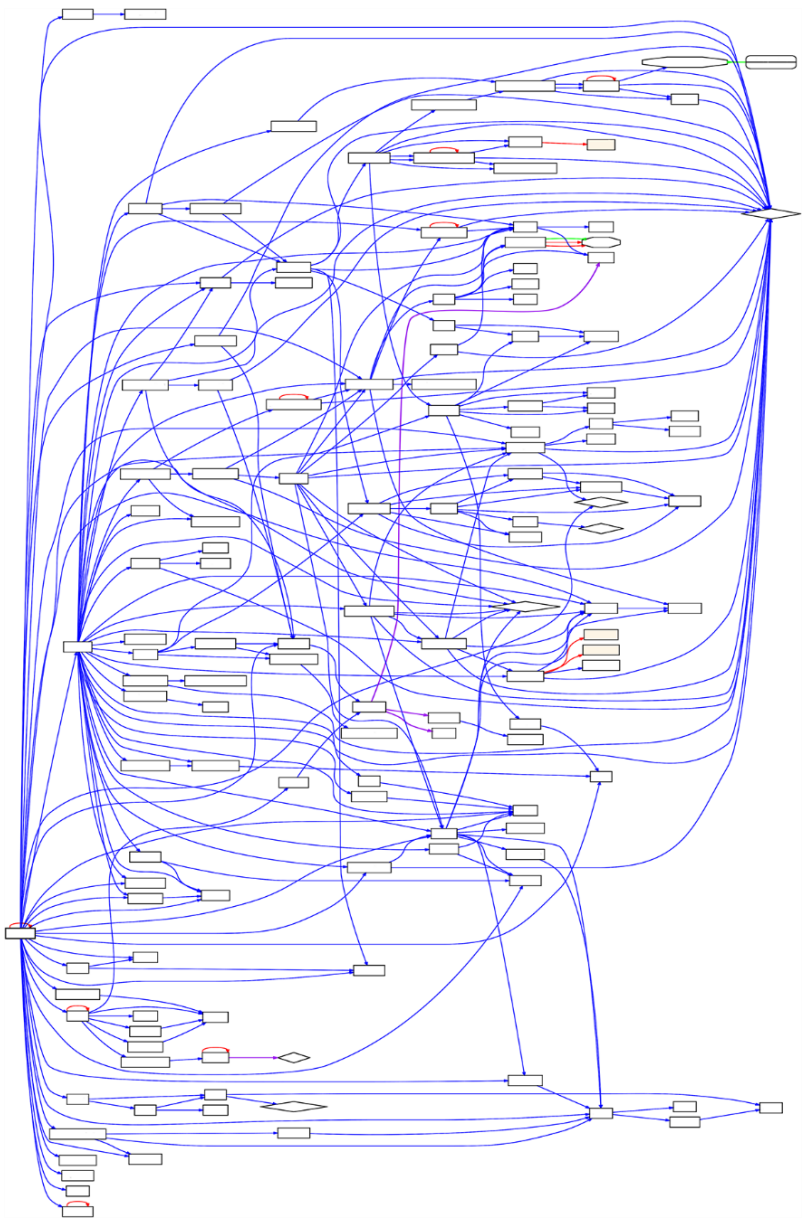


Figur 9. Beroendegrafen för OpenVPN.

Den primära förutsättningen för att OpenVPN ska kunna fungera är att det finns flera förinstallerade mjukvarubibliotek, enligt beroendena i figur 9. För OpenVPN krävs minst 22 andra mjukvaror och bibliotek för att den ska kunna fungera i denna Linux-distribution. Beroendegrafen kan se olika ut för olika Linux-distributioner då paketeringen av mjukvara görs på olika sätt.

I regel har större mjukvaruprojekt oftast fler beroenden till andra mjukvaror och bibliotek, medan mindre och enklare mjukvaror har färre beroenden. Som

jämförelse behöver webbläsaren Firefox över 100 mjukvarubibliotek, vilket illustreras i figur 10.



Figur 10. Beroendegraf för Firefox.

Utöver beroenden till olika mjukvarubibliotek kan mjukvaran behöva hjälpprogram eller så kallad rekommenderad mjukvara. I fallet med OpenVPN rekommenderas det exempelvis att även installera *easy-rsa*, ett verktyg för att skapa certifikat som i sin tur kan användas av OpenVPN. Verktyget är dock ingen förutsättning för att OpenVPN ska fungera.

Andra viktiga mjukvaruberoenden som inte inkluderats i exemplen är beroenden till bakomliggande mjukvaror i datorn, såsom operativsystem, drivrutiner och inbyggd mjukvara.

4.2.4 Utvecklare

Mjukvara utvecklas allt oftare av flera olika aktörer och levereras som en sammanhållen slutprodukt till användaren. Tabell 4 visar antalet utvecklare och deras respektive bidrag till de olika mjukvarorna i undersökningen vid den tidpunkt som datainsamlingen gjordes i fallstudien. I Notepad++ har 251 individuella utvecklare varit involverade, trots att mjukvaran till 62 % är utvecklad av en enda individ. I större mjukvaruprojekt, som Firefox med 7399 unika utvecklare, är det mer lika fördelat mellan delaktiga utvecklare. Detta beror på att en enda person rimligtvis inte skulle klara av att skriva en så pass omfattande mjukvara.

Många mjukvaror utvecklas och förändras konstant för att inkludera nya förbättringar. Hur ofta mjukvara blir uppdaterat kan ses i antalet ändringar (eng. commits), som är antalet enskilda tillfällen en utvecklare har skickat in förändringar i källkoden till projektet. I fallet med Firefox betyder det att sedan projektet skapades har det skickats in över 6 miljoner kodförändringar. Under perioden 2018-09-04–2019-09-04 skickades det in 56 289 ändringar, vilket motsvarar cirka 154 ändringar per dag.

En mer detaljerad analys av kodförändringarna i Firefox visar att det skickades in 5007 kodförändringar mellan den 5 augusti och 4 september 2019. Detta gjordes av 459 individer, varav 38 var nya utvecklare som aldrig tidigare har lämnat in kod till Firefox-projektet.

Tabell 4. Statistik över utvecklare för respektive mjukvara.

Mjukvara	Antal utvecklare	Enskilda utvecklares andelar[15]	Ändringar	Ändringar 12 månader[16]
Notepad++	263	62 %	3 479	608
OpenSSH	88	51 % 23 % 13 %	10 108	637
OpenVPN	135	28 % 11 % 10 %	5 947	357
Firefox	7 399	–	668 992	56 289
Chromium	8 102	–	810 704	104 657
Libreoffice	1 872	19 %	486 470	12 553

4.2.5 Reflektioner

Mjukvarans storlek korrelerar ofta med antalet utvecklare. Större projekt såsom webbläsare består av en stor mängd källkod i flera olika programmeringsspråk, vilket naturligt ger ett större behov av ett stort antal utvecklare. I större mjukvaruprojekt är det brukligt att genomföra en intern granskning av alla kodförändringar för att de ska tas in i det centrala kodförvaret (eng. repository), då en minimal kodförändring kan introducera sårbarheter.

En viktig aspekt av kvalitetskontroll och skydd mot manipulation är att källkoden testas och granskas på ett effektivt sätt. När ett projekt nått en stor kodbas, som exempelvis webbläsarna Firefox och Chromium, är det nästan omöjligt att kunna göra en heltäckande kodgranskning då komplexiteten blivit för hög. Med komplexiteten kommer även risken att en utvecklare kan dölja små men allvarliga sårbarheter i den källkod som denne bidrar med.

I större projekt används i regel flera olika programmeringsspråk för att implementera olika delar av mjukvaran. För att kunna granska koden behöver granskaren kunna förstå de olika programmeringsspråken och hur interaktion mellan dem fungerar. Källkod kan skrivas på olika sätt och med olika stilar, där

[15] Respektive andel av mjukvaran som skrivits av en enskild utvecklare. Tabellen tar endast upp utvecklare som var och en bidragit med minst 10 % av mjukvaran räknat i antal ändringar (eng. commits).

[16] Perioden avser de senaste 12 månaderna före indexeringstillfället i Open Hub. Se fotnot 12 för respektive indexeringsdatum.

exempelvis välskrivna kommentarer i källkoden kan underlätta för utomstående när de granskar koden.

För proprietär mjukvara är det i regel svårt att få reda på någon information om komplexiteten, exempelvis i form av antal källkodsrader, antalet utvecklare och respektive utvecklares bidrag till mjukvaran. En mjukvara som ligger nära de som har studerats är webbläsaren *Google Chrome*[17] som till stor del bygger på Chromium. Google Chrome har proprietära tillägg utöver Chromiums funktioner, vilket innebär att komplexiteten i Google Chrome sannolikt är högre än den för Chromium. Detta har dock inte gått att vare sig bekräfta eller vederlägga då informationen inte finns tillgänglig.

Microsoft Office är ytterligare ett exempel på proprietär mjukvara med liknande funktion som en av de undersökta mjukvarorna. Medan Libreoffice består av knappt tio miljoner rader källkod så anger en källa på internet att Microsoft Office 2013 består av 30–60 miljoner rader (Dvorak, 2013). Denna siffra kan dock inte bekräftas då den inte tycks ha publicerats av Microsoft.

Den viktigaste insikten kopplat till proprietär mjukvara är bristen på insikt. Eftersom källkod eller metainformation sällan finns tillgänglig går det exempelvis inte att veta hur koden ser ut, hur komplex den är och hur många som varit inblandade i utvecklingsarbetet. En fördel med öppen källkod är att källkoden finns tillgänglig för användaren och dennes organisation att granska. En sådan extern granskning är omöjlig för proprietära mjukvaror där källkoden inte finns att tillgå.

[17] https://www.google.com/intl/sv/chrome/

5. Hot mot cyberleveranskedjor

IT-system är i sig komplexa och hanteras dessutom i en cyberleveranskedja som i sin tur är komplex, vilket sammantaget medför att system kan angripas på många olika sätt. Angrepp i cyberleveranskedjan kan dessutom ske vid många olika tillfällen under leveranskedjans alla steg. Vidare sker utveckling och produktion globalt, vilket innebär att många olika företag och organisationer i åtskilliga länder deltar i utveckling och hantering av IT-systemen innan de når slutanvändaren.

William Evanina, chef för USA:s National Counterintelligence and Security Center (NCSC), menar att infiltration av leveranskedjan för mjukvara utgör ett av nyckelhoten som företag måste ta hänsyn till (Corera, 2018). Det är också något som bekräftas av Crowdstrikes undersökning från 2018, där 66 procent av de tillfrågade svarade att de någon gång utsatts för ett angrepp i leveranskedjan för mjukvara (Larson, 2018).

5.1 Generell hotbild

Enisa, EU:s byrå för nät- och informationssäkerhet, genomför ett omfattande och kontinuerligt arbete för att kartlägga den generella hotbilden mot IT-system. Arbetet har pågått under ett antal år och resulterat i flera rapporter. *Enisa threat landscape report 2018* tar upp femton hotområden inom cybersäkerhet. I rapporten lyfts angrepp i leveranskedjan som ett växande problem som företag och organisationer måste vara medvetna om (Sfakianakis, Douligeris, Marinos, Lourenco, & Raghimi, 2019, s. 8; Sfakianakis m.fl., 2019, s. 137). Angrepp i cyberleverantörskedjan kan vara effektiva sätt att påverka system och är ofta svåra att upptäcka, vilket kan göra dessa typer av angrepp attraktiva (Sfakianakis m.fl., 2019, s. 135).

Leverantörer av IT-system och delar till IT-system lever ofta med en ekonomisk press att snabbt ta fram nya produkter och sätta dem på marknaden. Samtidigt ska leverantörer möta kundernas förväntningar gällande funktionalitet och vara kostnadseffektiva för att kunna behålla eller utveckla sin marknadsposition. Detta leder till att angrepp i leveranskedjan måste anses vara en viktig del av cybersäkerhetsområdet (Sfakianakis m.fl., 2019, s. 135).

Angrepp mot cyberleveranskedjor kan användas i samband med cyberspionage. Angriparen kan exempelvis plantera olika funktioner i IT-systemen som underlättar för angriparen att komma in i systemen eller få ut information från dem. Cyberspionage är ett av de hotområden som tas upp av Enisa, där de även framhåller att angriparna i allt större grad utnyttjar brister hos leverantörer med svagare cybersäkerhet (Sfakianakis m.fl., 2019, s. 108).

Leveranskedjor kan användas av angripare för att nå bred spridning av skadlig kod, men även för att angripa en särskilt utvald organisation (Sfakianakis m.fl., 2019, s. 31).

En tidigare rapport från Enisa fokuserar på hot mot *fysiska produkter*[18] inom mobila och inbyggda tillämpningar (Vishik m.fl., 2017, s. 7). För denna typ av produkter tar Enisa upp följande generella hot (Vishik m.fl., 2017, s. 8):

- **Tillägg till eller modifiering av existerande hårdvara**. Ett exempel är Cottonmouth-1, ett implantat för USB-kablar som möjliggör trådlösa attacker mot enheten som USB-kabeln är ansluten till (Schneier, 2014). Andra exempel är olika typer av enheter som ansluts i interna eller externa gränssnitt, såsom Firewire och PCI Express.
- **Modifieringar av inbyggd mjukvara via befintliga uppdaterings-gränssnitt.** Ett exempel som tas upp är förändringar i *system management mode* på moderna processorer, vilket kan leda till bakdörrar som inte kan upptäckas av operativsystemet.
- **Angrepp mot inbyggd mjukvara**. Då inbyggd mjukvara kan innehålla samma typer av sårbarheter som vilken mjukvara som helst kan dessa utnyttjas vid ett angrepp. I detta fall tar Enisa upp ett exempel där en sårbarhet i den inbyggda mjukvaran på ett instickskort för Ethernet-nätverk gör att en angripare kan påverka nätverkstrafiken eller angripa datorn som kortet är monterat i.

Om slutanvändarorganisationen har ett högt säkerhetsmedvetande och bra rutiner för personalförsörjning kan det vara svårt för en antagonist att få in en insider i organisationen. I dessa lägen kan det vara enklare för antagonisten att infiltrera en leverantör med lägre säkerhetsmedvetande för att genomföra angreppet där. Precis som i andra sammanhang tenderar angripare att följa den enklaste vägen att nå sina mål, vilket innebär att angrepp i leveranskedjan inriktar sig på den svagaste länken i kedjan (Sfakianakis m.fl., 2019, s. 135).

5.2 Organisatoriska och administrativa problem

I en översikt om risker i cyberleveranskedjor identifierar Enisa att det inte finns några gemensamma riktlinjer för att mäta produktens integritet från produktion

[18] Enisas rapport använder termen *hardware* (hårdvara) men avser fysiska artefakter som består av både elektronik och inbyggd mjukvara. Eftersom deras diskussioner även inkluderar inbyggd mjukvara används termen *fysisk produkt* istället.

till driftsättning (Cadzow m.fl., 2015, s. 27). De poängterar att det finns riktlinjer och goda rutiner i flera branscher men att dessa inte alltid används konsekvent vid upphandling och inköp. Detta medför att det är svårt att säkerställa att produkterna inte blivit förändrade, utbytta eller felkonfigurerade (Cadzow m.fl., 2015, s. 19).

Ytterligare en rapport från Enisa tar upp ett antal risker och rekommendationer kopplat till upphandling och inköp av IT-produkter och IT-tjänster (Karsberg & Dekker, 2014). Några av de risker som lyfts fram, utöver misstag och insiderhot, är missförstånd kring faktisk kravbild runt säkerhetsfunktioner, oklar ansvarsfördelning vid incidenter, bristande styrning av leverantörens underleverantörer samt bristande kompetens hos leverantörens personal (Karsberg & Dekker, 2014).

Genom IT-systemens komplexitet blir det i praktiken omöjligt för slutanvändarorganisationer att undersöka systemen för att helt säkerställa att de inte blivit manipulerade, utbytta eller felkonfigurerade. Det saknas helt enkelt metoder och verktyg för att nå den graden av förtroende för systemet, vilket innebär att det blir extra viktigt att genomföra upphandlingar och inköp för att nå maximalt förtroende för produkterna som levereras.

5.3 Exempel på händelser

Under de senaste åren har det inträffat flera olika händelser som haft sin uppkomst under tiden som en leverantör ansvarat för systemet eller för en del av systemet. Nedan följer åtta exempel på sådana händelser.

5.3.1 Infekterad uppdatering av Ccleaner

Ccleaner är en mjukvara från företaget *Piriform* som används för att ta bort onödig eller oönskad data på Windows-datorer och som finns att ladda ner från företagets hemsida. Under 2017 drabbades Piriform av ett intrång i sitt nätverk. Vid angreppet ersattes den officiella nedladdningsbara versionen av Ccleaner på företagets webbplats med en infekterad version som spred skadlig kod till användarna (Khandelwal, 2018).

Det sannolika målet med angreppet var att stjäla information från stora internationella företag där mjukvaran användes. Trots att den infekterade versionen laddades ner över 2,2 miljoner gånger verkar inte angreppet ha lyckats nå målet (Khandelwal, 2018).

5.3.2 Övertagandet av Event-stream

Event-stream är ett välanvänt JavaScript-bibliotek publicerat som öppen källkod. Under 2018 hade biblioteket länge saknat aktiv utveckling och underhåll, när en person hörde av sig till bibliotekets upphovsman och anmälde sig som frivillig att ta hand om det (Riley, 2018). Upphovsmannen gav denne person administratörsrättigheter för biblioteket på samarbetsplattformen Github.

Det visade sig att personen egentligen genomförde ett angrepp genom social manipulation för att ta över Event-stream. När personen fått tillgång till biblioteket genomfördes en uppdatering som införde skadlig kod i biblioteket, avsedd att stjäla Bitcoin-plånböcker (Riley, 2018).

5.3.3 Google Play för att distribuera skadlig kod

Google Play är en tjänst för att publicera och hämta mobilapplikationer för Android-telefoner. Under 2017 upptäckte ett säkerhetsföretag att det fanns cirka 300 mobilapplikationer på Google Play som innehöll en specifik skadlig kod. Mobilapplikationerna hade laddats ner ett stort antal gånger innan detta upptäcktes. Uppskattningsvis infekterades 70 000 Android-enheter (Morse, 2017). Den skadliga koden gjorde telefonerna till deltagare i omfattande botnät som kunde användas till distribuerade överbelastningsattacker (eng. distributed denial of service, DDoS) mot webbplatser. Spridningen underlättades sannolikt av att många användare uppfattar Google Play som en pålitlig plats att hämta mobilapplikationer.

Det finns flera ytterligare exempel där mobilapplikationer med skadlig kod distribuerats via Google Play. Ett av dessa är *CamScanner*, en applikation med bra rykte och över 100 miljoner nedladdningar, som innehöll skadlig kod efter en uppdatering (Kaspersky Team, 2019).

5.3.4 M.E.Doc infekterat med Notpetya

Företaget *Intellect Service* i Ukraina är leverantör av mjukvaran *M.E.Doc*, ett bokföringsprogram som används hos en stor andel av de ukrainska företagen. Vid ett intrång i företagets nätverk i juni 2017 bytte angriparna ut en fil i uppdateringspaketet för M.E.Doc mot en fil som innehöll skadlig kod (Greenberg, 2018). Undersökningar efter intrånget på Intellect Services nätverk har visat att det skedde med hjälp av giltiga användaruppgifter och att angriparna kunde förändra uppdateringsservrarna delvis för att dessa inte uppdaterats på flera år (Cimpanu, 2017).

Den skadliga koden – en ransomware kallad *Notpetya* – spreds mycket snabbt och de datorer som drabbades blev oanvändbara. Förutom en omfattande spridning i Ukraina slogs datorsystem ut hos flera världomspännande företag, däribland de stora transportföretagen Maersk och TNT express (Greenberg, 2018).

5.3.5 Xcodeghost – Xcode med skadlig kod

Xcode är ett utvecklingsverktyg från Apple för att skapa mjukvara som ska köras under Apple Ios. Under 2015 upptäcktes det att en infekterad version av Xcode spreds via kinesiska webbplatser och att ett antal utvecklare laddat ner och använt denna version istället för att hämta verktyget från Apple (Rossignol, 2015). När den infekterade versionen av Xcode användes blev konsekvensen att även de utvecklade mobilapplikationerna förde med sig skadlig kod. Trots detta passerade mobilapplikationerna Apples granskning och distribuerades via App Store (Rossignol, 2015).

5.3.6 Left-pad borttaget från pakettjänsten Npm

Företaget *Npm* driver en pakethanteringstjänst med samma namn. Genom denna tjänst hanteras och distribueras ett stort antal mjukvarubibliotek för JavaScript. En konflikt mellan företaget och en utvecklare av öppen källkod ledde till att utvecklaren valde att ta bort all sin mjukvara från Npm-systemet, inklusive den enkla biblioteksfunktionen *left-pad* som bestod av cirka 10 rader källkod (Collins, 2016).

Funktionen användes dock direkt eller indirekt av många andra mjukvarupaket. Detta fick till följd att stora mängder annan mjukvara inte längre gick att använda när funktionen togs bort från Npm. På grund av de omfattande konsekvenserna valde Npm att själva återpublicera left-pad, trots att utvecklaren valt att ta bort den (Williams, 2016).

5.3.7 Bakdörr i mjukvaran på Xiaomi-telefoner

Android-telefonen *Xiaomi Mi4* innehöll en bakgrundsapplikation vid namn *AnalyticsCore* som skickade information om telefonen till företaget Xiaomi varje dygn men som även hämtade och installerade en uppdatering av applikationen om den fanns tillgänglig (Broenink, 2016).

Applikationen använde dessutom ett osäkert protokoll, HTTP, för kommunikationen med Xiaomis servrar (Broenink, 2016), vilket gjorde att eventuella angripare i nätverket kunde avlyssna eller manipulera informationen som skickas. Potentiellt medför detta även att en eventuell angripare hade

möjlighet att förfalska uppdateringar och exempelvis inkludera skadlig kod i dessa. Enligt Xiaomi ska detta dock inte vara möjligt då uppdateringarna måste vara signerade för att telefonen ska acceptera dem (Khandelwal, 2016).

5.3.8 Förfalskade elektronikkomponenter

Förfalskade elektronikkomponenter är vanligt förekommande (se Guin m.fl., 2014). Detta är särskilt vanligt för komponenter som gått ur produktion eller som är svåra att få tag på av andra orsaker. Välgjorda förfalskningar kan vara svåra att upptäcka och kan dessutom ge komplexa och svårdiagnosticerade problem i de system där de används (Pickering, 2017).

Myndigheterna i USA uppmärksammade tidigt förfalskade elektronik-komponenter som ett relevant och reellt problem. En undersökning från U.S. Department of Commerce, USA:s handelsdepartement, visade att 39 % av de organisationer och företag som tillfrågades hade påträffat förfalskade komponenter under tidsperioden 2005–2008 (Crawford m.fl., 2010). Under 2011 avslöjades att flera leverantörer till USA:s försvarsmakt hade lurats att använda förfalskade komponenter i sina produkter (Lemer, 2011). År 2012 släpptes en rapport från U.S. Government Accountability Office (GAO), USA:s motsvarighet till svenska Riksrevisionen, där de undersökte komponenter som köpts in via handelsplatser på internet. Slutsatsen var att samtliga 16 komponenter som köptes in i undersökningen var förfalskningar (Government Accountability Office, 2012a).

6. Principer för säkra leveranskedjor

Inom studien har ett antal rapporter som behandlar olika rekommendationer för att säkra upp cyberleveranskedjor identifierats. Detta kapitel presenterar ett urval av dessa rekommendationer, där samtliga kommer ur publikationer från *Nist*, *Enisa*, *Mitre* och *Safecode*. Urvalet syftar inte till att vara fullständigt, men ämnar ge en bild av vilka av dessa rekommendationer som denna rapports författare anser är viktigast. Utöver detta ges en sammanställning av ytterligare publikationer med råd. Den intresserade läsaren kan gå direkt till källornas mer detaljerade och mer kompletta vägledningar.

6.1 Metod

Inom studien har det genomförts en mindre litteraturstudie, fokuserad på att hitta dokument som tar upp metoder och principer för att arbeta med säkerhet i leveranskedjor för IT-produkter. Litteraturstudien baseras på explorativa sökningar i litteraturdatabaserna Scopus[19] och Google Scholar[20] samt på Google[21]. Detta resulterade i 17 publikationer med ett stort antal rekommendationer. Ur dessa har ett urval av rekommendationer från några välrenommerade aktörer gjorts. Rekommendationerna har sedan sammanställts till 15 principer.

En sammanfattning av innehållet i övriga publikationer har även utförts. Syftet med denna sammanfattning är att vägleda läsaren till ytterligare rekommendationer, exempelvis rörande tillverkning av hårdvarukomponenter.

6.2 Sammanställda principer

I publikationerna, som kommer från Nist, Mitre, Enisa och Safecode, återfinns ett stort antal rekommendationer för att skydda cyberleverantörskedjor. *Nist*[22] är en amerikansk myndighet som tagit fram två standarder relaterat till leveranskedjor. Publikationen NISTIR 7622 beskriver ett antal steg för att implementera riskhantering för leveranskedjor på en hög nivå (Boyens, Paulsen, Bartol, Shankles, & Moorthy, 2012). SP 800-161 presenterar en detaljerad

[19] https://www.scopus.com/

[20] https://scholar.google.se/

[21] https://www.google.com/

[22] https://www.nist.gov/

riskhanteringsprocess för leveranskedjor, det vill säga att den ger riktlinjer för hur risker identifieras, bedöms och mildras för leveranskedjan (Boyens, Paulsen, Moorthy, & Bartol, 2015).

Enisa[23] ger bland annat ut publikationer med rekommendationer. De har tagit fram en publikation och en artikel med rekommendationer kopplat till leveranskedjor (Cadzow m.fl., 2015; Enisa, 2018).

Mitre[24] är ett icke-vinstdrivande forskningsföretag som utför federalt finansierad forskning i USA. I en av Mitres publikationer presenteras kortsiktiga och långsiktiga åtgärder för att hantera leveranskedjor, omfattande allt från lagar till teknologi (Nissen, Gronager, Metzger, & Rishikof, 2018).

Safecode[25] är en icke-vinstdrivande organisation som publicerar rekommendationer framtagna av experter från industrin. Safecode har publicerat en rapportserie om hur tilltron säkerställs i leveranskedjan och som presenterar ett antal principer (Simpson, 2009).

Inom studien har en delmängd av dessa rekommendationer valts ut då de ansetts viktiga och ger en bred täckning vad gäller skydd av leveranskedjan. Rekommendationerna har sedan sammanställs till femton övergripande principer. De övergripande principerna är att

1. balansera åtgärder mot kostnader
2. identifiera alla aktörer, objekt och processer
3. välja samarbetspartner med omsorg
4. granska efterlevnad av regler och avtal
5. fastställa regler och avtal för samarbete
6. designa system, objekt och processer defensivt
7. säkerställa integritet under hela livscykeln
8. säkra upp utvecklingsmiljön
9. skapa spårbarhet i leveranskedjan
10. begränsa informationsspridning
11. skydda mot otillbörlig förändring
12. stärka leveransmekanismer
13. säkerställa integriteten vid uppdateringar
14. kassera information på rätt sätt
15. utbilda och skapa medvetenhet.

[23] https://www.enisa.europa.eu/

[24] https://www.mitre.org/

[25] https://safecode.org/

Följande avsnitt ger en mer detaljerad beskrivning av var och en av de övergripande principerna.

6.2.1 Balansera åtgärder mot kostnader

Målet med riskhantering är att finna en rimlig nivå av assurans att systemkomponenter inte blivit utbytta, manipulerade eller felkonfigurerade i leveranskedjan (Cadzow m.fl., 2015). Alla extra kontroller och mekanismer ökar priset på produkten och det ska därför alltid övervägas om produkten behöver det (Boyens m.fl., 2015).

Nist framhåller att endast de komponenter som innehåller programmerbar logik och är kritiska för systemfunktionaliteten bör utvärderas med hjälp av Nists standarder. Dessutom är inte varje komponent kritisk, oavsett om det är en integrerad krets, en router eller ett program. En analys bör göras för att utvärdera hur kritisk varje komponent anses vara, varefter resultatet bör användas för att bestämma vilka åtgärder som bör genomföras (Boyens m.fl., 2012).

6.2.2 Identifiera alla aktörer, objekt och processer

En förutsättning för att få kontroll på leveranskedjan är att veta vad och vem som finns i den och vilka processer som sätter samman den. Utan denna kunskap är det svårt att identifiera oönskade händelser som kan inträffa i leveranskedjan och därmed omöjligt att identifiera passande åtgärder. Nist föreslår att en identitetstruktur bör skapas med unikt identifierbara objekt, komponenter, verktyg, organisationer, personal, processer, kommunikationssätt och leveransmetoder (Boyens m.fl., 2012).

6.2.3 Välj samarbetspartner med omsorg

Förtroende för leverantörerna är en grundsten i leveranskedjan och därför bör dessa väljas med omsorg. Det finns ett antal kvalitetscertifieringar som kan användas vid bedömning, bland dessa finns ISO 9000-serien, ISO 27000-serien samt ISO 14000-serien. Exempelvis kan mognadsgraden vad gäller tekniska lösningar för integritet vägas in vid bedömningen, såsom att digitala signaturer används. Andra kriterier kan inkludera tidigare och nuvarande ekonomiska situation samt tidigare och nuvarande beteende och omdöme (Cadzow m.fl., 2015).

Det blir allt svårare att hålla en hög nivå av kontroll ju fler leverantörer som finns i leveranskedjan. Om det går att hålla nere antalet leverantörer blir det lättare att välja dessa med omsorg (Enisa, 2018).

6.2.4 Fastställ regler och avtal för samarbete

Krav på insyn i leveranskedjan kan inkluderas när ett samarbetskontrakt skrivs med en leverantör. På så sätt kan spårbarhet uppnås förbi leverantören. Krav kan ställas på att samtliga tredjepartskomponenter ska redovisas, oavsett om de licensierats under en proprietär eller öppen modell (Nissen m.fl., 2018). Krav kan också ställas på specifika tekniker som ska användas, exempelvis på specifika godkända protokoll (Enisa, 2018).

6.2.5 Granska efterlevnad av regler och avtal

Integratörer är en speciell kategori av leverantörer som sätter samman olika produkter från underleverantörer, ser till att de fungerar tillsammans och att de uppfyller kundens krav. Då kund och integratör ofta delar information och infrastruktur ställs krav på att dessa skyddas och behandlas på rätt sätt. Granskning av att integratören uppfyller de krav som är ställda i kontraktet behöver därför utföras så att eventuella åtgärder kan vidtas om kraven inte är uppfyllda. Verifiering kan exempelvis ske genom testning, övervakning och granskning. Det uppmuntras att integratören i sin tur granskar efterlevnad av regler och avtal med sina närmaste leverantörer (Boyens m.fl., 2012).

6.2.6 Designa system, objekt och processer defensivt

Robusthet mot angrepp i leveranskedjan bör byggas in redan vid design av system, objekt eller processer. I designen finns då förberedda planer för åtgärder vid oönskade händelser. En defensiv designstrategi ger ökad flexibilitet att hantera oönskade händelser och att anpassa sig till en förändrad miljö.

Ett exempel på defensiv design är att minska beroendet av produkter från en leverantör genom att köpa in likvärdiga produkter från flera tillverkare (Boyens m.fl., 2012). Värdet av att använda produkter från flera leverantörer måste dock vägas mot att leveranskedjan blir mer komplex och att det därmed kan bli svårare att upprätthålla kontrollen över leverantörerna, se avsnitt 6.2.3.

6.2.7 Säkerställ integritet under hela livscykeln

I hela produktens livscykel, från första designsteget fram till avveckling, behöver det finnas metoder eller processer för att säkerställa integritet i leveranskedjan. Integritetskontroller behövs för att upptäcka förändrade eller utbytta komponenter. I standarden ISO/IEC 15288 identifieras ett antal steg i livscykeln där det är viktigt att placera funktioner för att verifiera integriteten av produkten för leveranskedjan (Cadzow m.fl., 2015).

Några funktioner för att verifiera integriteten hos produkter kan handla om kodgranskning eller kodinspektion av mjukvara (Simpson, 2009). Stickprovskontroller kan användas för att värdera tilltron till produkterna. Frekvensen för sådana stickprov och vad som testas i dem behöver dock vara tillräckligt oförutsägbart för att förhindra att en angripare undgår detektion. Oberoende tredjepartsgranskning och certifiering kan även användas (Cadzow m.fl., 2015).

Eftersom även de mest analyserade, validerade och certifierade aktörer i leveranskedjan kan bli korrupta, måste integritetskontrollen också finnas inuti själva produkten. Trusted platform module (TPM) är ett exempel på en teknisk funktion som möjliggör detta genom att kontrollera mjukvarors riktighet vid inläsning. Det finns även metoder för att kontrollera integriteten hos hårdvarukomponenter (Cadzow m.fl., 2015).

6.2.8 Säkra upp utvecklingsmiljön

Integriteten hos systemet eller produkten kräver att leverantörens utvecklingsmiljö har erforderligt skydd för att försvåra otillbörlig förändring av produkten eller införsel av skadlig kod. Det kräver också att utveckling sker utifrån metoder för säker utveckling, inklusive test och verifiering (Cadzow m.fl., 2015). Enisa rekommenderar besök och inspektion av utvecklingsmiljöerna regelbundet (Enisa, 2018).

6.2.9 Skapa spårbarhet i leveranskedjan

Spårbarhet kan åstadkommas genom upprättande av register över objekt och deras ursprung samt historiken över vilka förändringar som skett av objekten och vem som gjort dem. Detta register bör upprätthållas gemensamt av inköpare och leverantörer (Boyens m.fl., 2012). Registret ger en tilltro till att varje förändring är auktoriserad, transparent och verifierbar (Simpson, 2009).

En metod bör inkluderas för att detektera falska spårbarhetsregister. Dock kan detta upplägg vara svårt eftersom många leverantörer arbetar under sekretessavtal. Detta innebär att de olika aktörerna i leveranskedjan inte kan undersöka spårbarhetsregistren för sina leverantörers leverantörer. Ett annat problem är att detta kräver att alla i leveranskedjan går med på att samarbeta kring spårbarhet för att skapa en förtroendekedja (Cadzow m.fl., 2015).

6.2.10 Begränsa informationsspridning

Inköpare och leverantörer behöver utbyta information för att kunna samarbeta. Information kan exempelvis inkludera design, leveransmetoder, driftsinstallation och hotbild. Informationen måste finnas tillgänglig där den behövs, men bör

samtidigt spridas med varsamhet. Regler för hantering av information i leveranskedjan bör därmed upprättas (Boyens m.fl., 2012).

Endast personer som behöver kritisk informationen för sin tjänst bör ha behörighet till denna. Skydd av kritisk information bör följa med informationen så att skyddet fortsätter att vara effektivt även om informationen flyttas (Simpson, 2009).

6.2.11 Skydda mot otillbörlig förändring

Behörigheter bör begränsas till det som krävs för att aktörerna i leveranskedjan ska kunna utföra sina uppgifter, exempelvis när det gäller att förändra objekt, processer, organisation, kommunikation och system. Det är viktigt att definiera tydliga roller med tydliga behörigheter för att skydda produkterna mot negativa konsekvenser (Boyens m.fl., 2012; Simpson, 2009). Obehöriga försök att förändra produkterna ska förhindras och, om detta misslyckas, så ska det vara enkelt att upptäcka förändringarna och rätta till dem (Simpson, 2009).

6.2.12 Stärk leveransmekanismer

Leveranser kan ske både fysiskt och digitalt, under flera skeenden i leveranskedjan och till flera aktörer. Leveransmetoderna bör därför stärkas för att skydda leveranserna mot otillbörlig påverkan på produkterna, såsom att en antagonist byter ut eller förändrar leveransen (Boyens m.fl., 2012).

6.2.13 Säkerställ integriteten vid uppdateringar

Vidmakthållande av en produkt eller system startar vid driftsättning och slutar vid avveckling. Under hela vidmakthållandet måste integriteten säkerställas när förändringar behöver utföras. Detta gäller vid förändringar och uppdateringar som rör allt från mjukvara, hårdvara och utbyteskomponenter till driftprocesser. Möjligheterna att påverka säkerhetsegenskaperna hos objekt och driftsprocesser ska minimeras (Boyens m.fl., 2012).

6.2.14 Kassera information på rätt sätt

Under hela livscykeln och i leveranskedjan kan information behöva kasseras. Information kan finnas på papper eller i digitala format, men kan även utgöras av hårdvara eller verktyg. Kassationen bör ske på lämpligt sätt så att informationen inte riskerar att hamna i fel händer och därmed ge insikt om produkten eller systemet som kan leda till otillbörlig tillgång (Boyens m.fl., 2012).

6.2.15 Utbilda och skapa medvetenhet

Utbildning och träning krävs för att höja medvetenheten hos aktörerna i leveranskedjan kring de risker som finns och hur dessa kan hanteras. Utbildningen bör täcka gällande policyer, procedurer, rutiner och tekniska lösningar för att minska risker i leveranskedjan (Boyens m.fl., 2012).

6.3 Mer läsning från övriga publikationer

Under litteraturstudien fann vi flera publikationer som beskriver hur IT-säkerhetsproblematik i leveranskedjan kan hanteras. Dessa har inte utgjort underlag för principerna, men rekommenderas som vidare läsning. I detta avsnitt ges en kort presentation av dessa publikationer.

År 2015 arrangerade Nist en workshop för att diskutera forskning kopplat till säkra leveranskedjor. Workshopen baserades på arton fallstudier som Nist genomfört tillsammans med företag från industrin i syfte att ta reda på deras rekommendationer för säkra leveranskedjor (National Institute of Standards and Technology, 2016). Materialet från denna workshop finns tillgängligt på Nist:s webbsida och innehåller frågeställningar kring leveranskedjor och rekommendationer på åtgärder. I materialet poängteras att det finns ett stort antal standarder och ramverk för säkra leveranskedjor. Det finns även en tabell med en sammanställning över sådana standarder och ramverk.

HKCERT[26] (2018) föreslår ett antal åtgärder, både tekniska och organisatoriska, flera liknande de som tas upp i principerna i avsnitt 6.2. GAO ger övergripande rekommendationen att ta fram policyer, processer och metoder för uppföljning (Government Accountability Office, 2012b). Ett dokument från Darpa problematiserar kring leveranskedjor för elektronik och föreslår åtgärder för att förhindra och detektera otillbörlig påverkan samt vilket agerande som är lämpligt vid upptäckt av otillbörlig påverkan (Defense Advanced Research Projects Agency, u.å.). Även ISA[27] har rekommendationer för att säkerställa integriteten hos elektronik i leveranskedjan (Borg, 2013). Det australiska försvarsdepartementet föreslår åtgärder för att detektera och förhindra trojaner i hårdvara (Beaumont, Hopkins, & Newby, 2011). I en rapport av Gansler m fl. (2014) ges förslag på åtgärder för att hantera förfalskade komponenter hos försvarsdepartementet i USA.

[26] Hong Kong Computer Emergency Response Team

[27] Internet Security Alliance

Fyra forskningsartiklar tar upp aspekter kring riskhantering för leveranskedjor. Alberts, Dorofee, Creel, Ellison och Woody (2011) behandlar hur risk kan värderas. Windelberg (2016) analyserar vilka målen är med riskhantering. Edwards, Kao, Hamlet och Bailon (2016) samt Ellison och Woody (2010) föreslår metoder för hur riskhantering kan genomföras. En rapport av Ellison, Alberts, Creel, Dorofee och Woody (2010) behandlar riskhantering avseende både produkter och system av system. En forskningsartikel föreslår en metod för hur leverantörer kan utvärderas (Qiu, Cao, Li, & Zhao, 2013).

6.4 Reflektioner

De femton principer som lyfts fram täcker in aktiviteter som spänner över hela leveranskedjan. Principerna ger inte en heltäckande lösning på problemet, men ger några handlingspunkter att börja med. Det handlar i grunden om risk-hantering och det krävs analyser för att utröna vilka risker som kan accepteras och vilka som behöver hanteras. Genom ett strukturerat analysarbete går det att lägga den ekonomiska tonvikten där den bäst behövs för att mildra risker i leveranskedjan. Eftersom det i de flesta fall inte finns ekonomiska förutsättningar för att säkra alla delar av leveranskedjan för alla IT-system.

Grunden sätts genom att skapa en situationsbild över vilka leverantörer, objekt och processer som ingår i leveranskedjan. Att rita upp hur leveranskedjan hänger samman är en förutsättning för att kunna ta reda på vilka risker som finns eller vilka potentiella risker som kan uppstå.

Risker kan vara interna, dvs uppstå från verksamheten inom organisationen. De kan också vara externa för organisationen men inom leveranskedjan. Den miljö som leveranskedjan är exponerad mot kan förändras vilket kan skapa nya risker. Exempelvis kan lagstiftningen ändras så att en produkt tvingas innehålla bakdörrar. Ändrade omständigheter som exempelvis krig eller politiska beslut, kan förändra förutsättningarna för att samarbeta med ett specifikt land.

Identifiering av risker i leveranskedjan innebär att identifiera oönskade händelser som kan uppstå i hela leveranskedjan. Det är dock inte alltid möjligt att täcka in hela leveranskedjan beroende på dess storlek och komplexitet, vilket kan vara fallet vid produktion av merparten av mjukvaror och hårdvaror. Aktiviteten ger dock en täckning som är bättre än att inte utföra någon riskanalys alls. För de identifierade riskerna, ska slutligen riskhanteringsbeslut tas om risken kan accepteras eller om åtgärder krävs. Åtgärder kan vara sådana som presenterats i form av principer i detta kapitel, exempelvis att etablera spårbarhet och säkerställa integritet eller att stärka leveransmekanismer. Om en produkt är tillverkad av en leverantör som det saknas tillit för, skulle en möjlig

åtgärd kunna vara att byta till en annan mer lämplig leverantör. En lämplighets-analys bör utföras för att utröna om varje ny leverantör är att betrakta som en lämplig samarbetspartner. Lämpligt utformade avtal behöver etableras för att reglera hur samarbetet ska gå till. Dessutom är utbildning i de regler och processer som hanterar leveranskedjeproblematik något som behöver genomföras, både internt och hos samarbetspartners. En medvetenhet om leveranskedjerisker är viktig för att åtgärder som processer och regler ska efterlevas.

Genom ett strukturerat riskhanteringsarbete kan systemen redan från början designas för att vara förberedda och ha en resiliens mot angrepp av aktörer i leveranskedjan.

7. Diskussion

Risker kopplade till cyberleveranskedjor är generellt sett mycket svårhanterade, oavsett vilka typer av produkter det gäller. Eftersom produkterna är utanför mottagarens kontroll under (i stort sett) hela leveranskedjan minskar dennes möjligheter att påverka produkternas egenskaper. I många fall handlar leveranskedjesäkerhet huvudsakligen om att *rätt mängd* produkter med *tillräcklig minsta kvalitetsnivå* levereras. För komplexa produkter blir kvalitetsnivå ett komplicerat begrepp, som omfattar många olika aspekter rörande hållbarhet och funktion.

Avsiktlig, mer eller mindre antagonistisk påverkan på produkter kan förkomma inom många olika branscher, där skandalen med manipulerad mjölkersättning i Kina är ett tydligt exempel. En kemikalie, melamin, hade tillsatts i mjölken för att analyserna skulle visa en högre proteinhalt än mjölken egentligen hade (Jacobs, 2008). Problemet visade sig först när många spädbarn blev sjuka och några av dem dog på grund av melaminets verkningar. Drivkraften bakom händelsen var troligen ekonomisk, då en till synes bättre produkt är lättare att sälja och ger högre avkastning.

När drivkraften snarare är antagonistisk *mot slutanvändaren* kan påverkan på produkterna vara än mer subtil och välanpassad. Genom sin omfattande komplexitet kan IT-system lätt dölja både avsiktliga och oavsiktliga sårbarheter som kan nyttjas av antagonister. Latenta sårbarheter kan ligga dolda under lång tid för att sedan aktiveras när det passar angriparen bäst.

Komplexiteten hos IT-systemen är IT-säkerhetens största fiende, då det är denna som gör det otroligt svårt att verifiera ett system i sådan utsträckning att det garanterat är fritt från sårbarheter. Som vi sett i fallstudierna i kapitel 4 är komplexiteten stor för både hårdvara och mjukvara när det gäller såväl leveranskedja som teknisk lösning. Webbläsare är ett exempel på en vanligt förekommande mjukvara med stor potentiell säkerhetspåverkan eftersom de har samtidig tillgång till alla faror på internet och till en stor del av datorn. Både Firefox och Chromium består av tiotals miljoner rader källkod från tusentals utvecklare, vilket gör det sannolikt att det någonstans i detta finns kritiska sårbarheter som angripare kan nyttja. Komplexiteten hos systemen innebär att IT-säkerhetsarbetet i praktiken måste utgå från att alla IT-system innehåller sårbarheter, eftersom dessa är så pass frekvent förekommande i mjukvara.

I IT-säkerhetssammanhang talas det ofta om *lökprincipen*, det vill säga att systemet bör skyddas genom många oberoende lager av säkerhetsåtgärder. I praktiken bildar säkerhetsåtgärderna dock inte en lök – där varje lager i stort omsluter hela insidan – utan en vitkål, där varje säkerhetsåtgärd täcker en del av problemet och den sammanlagda mängden av åtgärder ger *tillräckligt många*

lager skydd med tillräckligt överlapp. I praktiken räcker det inte med att granska och verifiera det slutliga systemet då komplexiteten i regel är för hög för att det ska gå att fånga alla tänkbara sårbarheter. Därför måste en del av vitkålen skydda systemet mot risker från leveranskedjan genom att minimera möjligheterna att negativt påverka systemet redan i leverantörsleden.

Kapitel 5 visar att det finns en generell hotbild mot cyberleveranskedjan och att det har inträffat ett antal olika angrepp med varierande finess och olika följdverkningar. Detta innebär att alla IT-system utsätts för ett grundhot genom leveranskedjan som måste hanteras med hjälp av olika metoder, vilka sammantaget utgör en strategi för att hantera riskerna. Riskhanteringen kan bli mycket omfattande för system med kritisk funktion eller kraftig hotbild. Hot kan realiseras på mycket sofistikerade sätt i situationer där hotet kommer från resursstarka aktörer, såsom stater, och riktar sig mot högintressanta målsystem, exempelvis inom kritisk infrastruktur eller militärt försvar. Som vanligt finns inte ett givet sätt att sköta riskhantering, utan detta måste utgå ifrån systemets förutsättningar och hur viktigt systemet är för verksamheten.

Riskhantering vid upphandling och inköp av IT-produkter blir lätt en komplicerad uppgift med många faktorer som påverkar resultatet. Vem som har systemansvar och hur avtalen har formulerats är två av dessa faktorer. En avtalsmässig fråga som direkt kopplar till systemansvar är om slutanvändarorganisationen själv får åtgärda sårbarheter och andra problem i systemen eller om dessa måste hanteras av leverantören, exempelvis på grund av ansvarsklausuler och garantiåtaganden i avtalen.

Försvarsmaktens IT-system utgår i stor utsträckning från samma produkter som återfinns i IT-system inom andra organisationer. Dessa produkter utvecklas och tillverkas på en internationell marknad, med komplexa leveranskedjor som följd. Händelserna som tas upp i avsnitt 5.3 visar på några olika typer av sårbarheter som uppstått i cyberleveranskedjor. Dessa händelser illustrerar en grundhotnivå som Försvarsmakten måste ta hänsyn till vid exempelvis inköp och uppdatering av IT-system. Det är värt att notera att de flesta av händelserna bygger på relativt enkla angrepp mot leverantörer eller deras IT-system för att kunna påverka produkterna. En mer sofistikerad hotagent med större resurser kan antas ha möjlighet att genomföra betydligt mer avancerade och subtila angrepp mot leverantörerna för att nå sina mål. Detta gör säkerheten i cyberleveranskedjan ännu mer relevant för en organisation som Försvarsmakten.

Denna studie har undersökt cyberleveranskedjor i allmänhet med utgångspunkt i publicerade händelser och principer som inte är specifika för en militär kontext. Försvarsmakten har i många fall extra höga krav på IT-säkerheten i sina system och utsätts för hot av mycket starka antagonister. Detta innebär att det finns god

anledning att gå vidare med en undersökning av den speciella problemställning som gäller kring leveranskedjor för Försvarsmaktens IT-system

Ett balanserat IT-säkerhetsarbete är en central del i skyddet av information och verksamheter som är beroende av IT-system. Balansen mellan skyddsåtgärder och kostnader är också en av de principer för säkra leveranskedjor som tas upp i kapitel 6. Det som särskiljer säkerhet i cyberleveranskedjor mot normalt IT-säkerhetsarbete är att organisationen har avsevärt lägre grad av kontroll över leveranskedjorna jämfört med organisationens inflytande över driftmiljöerna. Många av principerna fokuserar därför helt naturligt på avtalsfrågor och avtalsefterlevnad, då dessa är de styrmedel som huvudsakligen finns att tillgå när det gäller att påverka hur leverantörerna hanterar produkterna.

Riskhantering och riskminimering är de huvudsakliga åtgärder som finns för att hantera de säkerhetsproblem som uppstår i IT-systemens leveranskedjor. När gränsen är nådd för vilka effekter som riskhanteringen kan ge återstår i praktiken bara att förlita sig på leverantören och att denne faktiskt levererar produkter utan antagonistiska sårbarheter, skadlig kod eller andra oönskade faror. Lågt försäljningspris är sällan ett incitament för leverantören att satsa på säkerhetsåtgärder, då det ofta är viktigare att funktionen finns på plats än att produkten är säker för köparen. Därmed bör högt förtroende för leverantören och produkten vara en minst lika viktig utvärderingsfaktor vid upphandling som lågt pris.

8. Referenser

Alberts, C. J., Dorofee, A. J., Creel, R., Ellison, R. J., & Woody, C. (2011). A Systemic Approach for Assessing Software Supply-Chain Risk. *2011 44th Hawaii International Conference on System Sciences*, 1–8. https://doi.org/10.1109/HICSS.2011.36

Beaumont, M., Hopkins, B., & Newby, T. (2011). *Hardware Trojans—Prevention, Detection, Countermeasures (A Literature Review)* (s. 50). Austalian Government, Department of Defence, Defence Science and Technology Organisation.

Borg, S. (2013). *The ISA Guidelines for Securing the Electronics Supply Chain* (s. 58). Internet Security Alliance.

Boyens, J., Paulsen, C., Bartol, N., Shankles, S. A., & Moorthy, R. (2012). *Notional Supply Chain Risk Management Practices for Federal Information Systems* (Nr NIST IR 7622; s. NIST IR 7622). https://doi.org/10.6028/NIST.IR.7622

Boyens, J., Paulsen, C., Moorthy, R., & Bartol, N. (2015). *Supply Chain Risk Management Practices for Federal Information Systems and Organizations* (Nr NIST SP 800-161; s. NIST SP 800-161). https://doi.org/10.6028/NIST.SP.800-161

Broenink, T. (2016, september 13). Reverse Engineering Xiaomi's Analytics app | Thijs Broenink. Hämtad 08 november 2019, från Thijs Breonink website: http://blog.thijsbroenink.com/2016/09/xiaomis-analytics-app-reverse-engineered/

Cadzow, S., Giannopoulos, G., Merle, A., Storch, T., Vishik, C., Gorniak, S., & Ikonomou, D. (2015). *Supply Chain Integrity: An overview of the ICT supply chain risks and challenges, and vision for the way forward*. Hämtad från Enisa website: https://www.enisa.europa.eu/publications/sci-2015

Cimpanu, C. (2017, juli 6). M.E.Doc Software Was Backdoored 3 Times, Servers Left Without Updates Since 2013. Hämtad 08 november 2019, från Bleeping Computer website: https://www.bleepingcomputer.com/news/security/m-e-doc-software-was-backdoored-3-times-servers-left-without-updates-since-2013/

Collins, K. (2016, mars 27). How one programmer broke the internet by deleting a tiny piece of code. Hämtad 08 november 2019, från Quartz website: https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code/

Corera, G. (2018, juli 26). US warns of supply chain cyber-attacks. Hämtad 08 november 2019, från https://www.bbc.com/news/technology-44941875

Crawford, M., Telesco, T., Nelson, C., Bolton, J., Bagin, K., & Botwin, B. (2010). *Defense Industry Base Assessment: Counterfeit Electronics*. Hämtad från U.S. Department of Commerce website: https://www.bis.doc.gov/index.php/documents/technology-evaluation/37-defense-industrial-base-assessment-of-counterfeit-electronics-2010/file

Defense Advanced Research Projects Agency. (u.å.). *A DARPA Approach to Trusted Microelectronics*. Hämtad från https://www.darpa.mil/attachments/ATrustthroughTechnologyApproach_FINAL.PDF

Dullien, T. (2018). *Security, Moore's law, and the anomaly of cheap complexity*. Keynote presenterad vid 10th International Conference on Cyber Conflict. Hämtad från https://www.youtube.com/watch?v=q98foLaAfX8

Dvorak, J. C. (2013). Microsoft Office's Spaghetti Code Mess. Hämtad 14 november 2019, från PC Magazine Opinion website: https://uk.pcmag.com/opinion/15915/microsoft-offices-spaghetti-code-mess

Edwards, N. J., Kao, G. K., Hamlet, J. R., & Bailon, J. (2016). *Supply Chain Decision Analytics: Application and Case Study for Critical Infrastructure Security*. 8.

Ellison, R., Alberts, C., Creel, R., Dorofee, A., & Woody, C. (2010). *Software Supply Chain Risk Management: From Products to Systems of Systems* (s. 45). Software Engineering Institute.

Ellison, R., & Woody, C. (2010). Supply-Chain Risk Management: Incorporating Security into Software Development. *2010 43rd Hawaii International Conference on System Sciences*, 1–10. https://doi.org/10.1109/HICSS.2010.355

Enisa. (2018, oktober 25). Supply Chain Attacks Back on the Agenda [Cyber security info note]. Hämtad 08 november 2019, från Enisa website: https://www.enisa.europa.eu/publications/info-notes/supply-chain-attacks-back-on-the-agenda

Gansler, J. S., Lucyshyn, W., & Rigilano, J. (2014). *Addressing Counterfeit Parts in the DoD Supply Chain:* https://doi.org/10.21236/ADA613231

Government Accountability Office. (2012a). *DoD Supply Chain: Suspect Counterfeit Electronic Parts Can Be Found on Internet Purchasing Platforms*. Hämtad från United States Government Accountability Office website: https://www.gao.gov/assets/590/588736.pdf

Government Accountability Office. (2012b). *IT Supply Chain: National Security-Related Agencies Need to Better Address Risks*. Hämtad från United States Government Accountability Office website: https://www.gao.gov/assets/590/589568.pdf

Greenberg, A. (2018, augusti 22). The Untold Story of NotPetya, the Most Devastating Cyberattack in History. Hämtad 08 november 2019, från Wired website: https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/

Guin, U., Huang, K., DiMase, D., Carulli, J. M., Tehranipoor, M., & Makris, Y. (2014). Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain. *Proceedings of the IEEE*, *102*(8), 1207–1228. https://doi.org/10.1109/JPROC.2014.2332291

HKCERT. (2018). HKCERT - Security Guideline. Hämtad 18 november 2019, från HKCERT website: https://www.hkcert.org/my_url/en/guideline/18041201

Hughes, P. (2017, februari 27). Inside the numbers: 100 billion ARM-based chips. Hämtad 08 november 2019, från Arm Community website: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/inside-the-numbers-100-billion-arm-based-chips-1345571105

Ingemarsdotter, J., Eidenskog, V., & Hedtjärn Swaling, V. (under utgivning). *Vilse i lasagnen? En upptäcktsfärd i den svenska digitaliseringens mångbottnade problemstruktur*. Totalförsvarets forskningsinstitut (FOI).

Jacobs, A. (2008, december 2). Chinese Release Increased Numbers in Tainted Milk Scandal. *The New York Times*. Hämtad från https://www.nytimes.com/2008/12/03/world/asia/03milk.html

Karsberg, C., & Dekker, M. (2014). *Security Guide for ICT Procurement: ICT Procurement Security Guide for Electronic Communications Service Providers*. Hämtad från Enisa website: http://dx.doi.org/10.2824/994989

Kaspersky Team. (2019, augusti 27). CamScanner is a malicious Android app with more than 100 million downloads in Google Play. Hämtad 08 november 2019, från Kaspersky official blog website: https://www.kaspersky.com/blog/camscanner-malicious-android-app/28156/

Khandelwal, S. (2016, september 15). Xiaomi Can Silently Install Any App On Your Android Phone Using A Backdoor. Hämtad 08 november 2019, från The Hacker News website: https://thehackernews.com/2016/09/xiaomi-android-backdoor.html

Khandelwal, S. (2018, april 18). CCleaner Attack Timeline—Here's How Hackers Infected 2.3 Million PCs. Hämtad 08 november 2019, från The Hacker News website: https://thehackernews.com/2018/04/ccleaner-malware-attack.html

Larson, D. (2018, juli 23). Global Survey Reveals Supply Chain as a Rising and Critical New Threat Vector. Hämtad 08 november 2019, från Crowdstrike blog website: https://www.crowdstrike.com/blog/global-survey-reveals-supply-chain-as-a-rising-and-critical-new-threat-vector/

Lemer, J. (2011, november 8). US military duped into using counterfeit parts. *Financial Times*. Hämtad från https://www.ft.com/content/adab0c50-0a3a-11e1-85ca-00144feabdc0

McCandless, D., Doughty-White, P., & Quick, M. (2015, september 24). Million Lines of Code—Information is Beautiful. Hämtad 08 november 2019, från Information is beautiful website: https://informationisbeautiful.net/visualizations/million-lines-of-code/

Moore, G. E. (2006a). Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. *IEEE Solid-State Circuits Society Newsletter*, *11*(3), 33–35. https://doi.org/10.1109/N-SSC.2006.4785860

Moore, G. E. (2006b). Progress in digital integrated electronics [Technical literaiture, Copyright 1975 IEEE. Reprinted, with permission. Technical Digest. International Electron Devices Meeting, IEEE, 1975, pp. 11-13.]. *IEEE Solid-State Circuits Society Newsletter*, *11*(3), 36–37. https://doi.org/10.1109/N-SSC.2006.4804410

Morse, J. (2017, augusti 30). Beware, Google Play Store gets caught distributing malware. Hämtad 08 november 2019, från Mashable website: https://mashable.com/2017/08/29/google-play-store-malware-wirex-botnet/

National Institute of Standards and Technology. (2016). Industry Best Practices: Cyber Supply Chain Risk Management. Hämtad 21 november 2019, från CSRC website: https://csrc.nist.gov/Projects/cyber-supply-chain-risk-management/Best-Practices

Nissen, C., Gronager, J., Metzger, R., & Rishikof, H. (2018). *Deliver Uncompromised: A Strategy for Supply Chain Security and Resilience in Response to the Changing Character of War* (s. 56). Hämtad från Mitre website: https://www.mitre.org/publications/technical-papers/deliver-uncompromised-a-strategy-for-supply-chain-security

Pickering, P. (2017, oktober 3). The Distribution Channel And The Fight Against Counterfeit Components. Hämtad 08 november 2019, från Design World website: https://www.designworldonline.com/the-distribution-channel-and-the-fight-against-counterfeit-components/

Qiu, X., Cao, L., Li, P., & Zhao, L. (2013). A Trust Evaluation Method for Supplier Selection. *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 1498–1503. https://doi.org/10.1109/TrustCom.2013.182

Qualcomm. (2017, november 8). Qualcomm Datacenter Technologies Announces Commercial Shipment of Qualcomm Centriq 2400 – The World's First 10nm Server Processor and Highest Performance Arm-based Server Processor Family Ever Designed. Hämtad 08 november 2019, från Qualcomm website: https://www.qualcomm.com/news/releases/2017/11/08/qualcomm-datacenter-technologies-announces-commercial-shipment-qualcomm

Riley, D. (2018, november 26). Bitcoin-stealing code inserted into popular GitHub-hosted JavaScript library. Hämtad 08 november 2019, från SiliconANGLE website: https://siliconangle.com/2018/11/26/bitcoin-stealing-code-inserted-popular-github-hosted-javascript-library/

Rossignol, J. (2015, september 20). What You Need to Know About iOS Malware XcodeGhost—MacRumors. Hämtad 08 november 2019, från MacRumors website: https://www.macrumors.com/2015/09/20/xcodeghost-chinese-malware-faq/

Schneier, B. (2014, mars 5). COTTONMOUTH-I: NSA Exploit of the Day—Schneier on Security. Hämtad 08 november 2019, från Schneier on Security website: https://www.schneier.com/blog/archives/2014/03/cottonmouth-i_n.html

Sfakianakis, A., Douligeris, C., Marinos, L., Lourenco, M., & Raghimi, O. (2019). *ENISA Threat Landscape Report 2018: 15 Top Cyberthreats and Trends*. Hämtad från Enisa website: http://dx.doi.org/10.2824/622757

Simpson, S. (2009). *The Software Supply Chain Integrity Framework: Defining Risks and Responsibilities for Securing Software in the Global Supply Chain*. Hämtad från Safecode website: http://safecode.org/publication/SAFECode_Supply_Chain0709.pdf

Vishik, C., Oswald, D., Sigl, G., Bratus, S., Pendarakis, D., Touzeau, J., & Klasen, W. (2017). *Hardware Threat Landscape and Good Practice Guide*. Hämtad från Enisa website: https://www.enisa.europa.eu/publications/hardware-threat-landscape

Williams, C. (2016, mars 23). How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript. Hämtad 08 november 2019, från The Register website: https://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/

Windelberg, M. (2016). Objectives for managing cyber supply chain risk. *International Journal of Critical Infrastructure Protection*, *12*, 4–11. https://doi.org/10.1016/j.ijcip.2015.11.003

Denna studie undersöker vad cyberleveranskedjor är och vilka hotbilder som finns mot dessa. Syftet med studien är att förmedla kunskap gällande de risker som kan uppstå i cyberleveranskedjor och vad som går att göra för att motverka dessa risker. Kunskapen är avsedd att underlätta diskussioner kring cyberleveranskedjor och de risker som de medför för IT-system i Försvarsmakten.

Rapporten riktar sig huvudsakligen till personer som arbetar med anskaffning, utveckling och förvaltning av Försvarsmaktens IT-system.

Inom studien genomfördes två fallstudier för att belysa komplexiteten i cyberleveranskedjor för både hårdvara och mjukvara. Fallstudiernas resultat visar att många olika aktörer är inblandade i cyberleveranskedjorna och att de är globala i sin omfattning.

Studien visar att det inte finns någon enkel lösning för att säkra upp cyberleveranskedjor. I stället krävs en gedigen riskhantering som tar hänsyn till hela kedjan. Målet med riskhanteringen är att få en resiliens mot antagonistiska hot i cyberleveranskedjan. Studien presenterar femton viktiga principer med konkreta handlingspunkter som är sammanställda från publikationer av Nist, Enisa, Mitre och Safecode. Principerna utgör ingen fullständig lista över vad som behöver utföras, men ger en utgångspunkt med några aktiviteter som är särskilt viktiga att beakta.

FOI

ISSN 1650-1942

www.foi.se