FOI

Litteraturstudie av tekniker för pålitliga IT-plattformar

JACOB LÖFVENBERG, IOANA RODHE

FOI-R--3724--SE
ISSN 1650-1942 September 2013

Jacob Löfvenberg, Ioana Rodhe

Litteraturstudie av tekniker för pålitliga IT-plattformar

Titel	Litteraturstudie av tekniker för pålitliga IT-plattformar
Title	Literature Review of Trusted Platform Techniques
Rapportnummer	FOI-R--3724--SE
Månad	September
Utgivningsår	2013
Antal sidor	40
ISSN	ISSN 1650-1942
Uppdragsgivare	Försvarsmakten
Projektnummer	E36047
Godkänd av	Christian Jönsson
Ansvarig avdelning	Informations- och aerosystem

FOI Totalförsvarets forskningsinstitut

Sammanfattning

I dagens mycket stora mjukvarusystem är det vanligt att tillgängliggöra uppdateringar av mjukvaran efter hand som fel korrigeras och funktionalitet förbättras. I verksamheter med höga sekretesskrav och/eller tillgänglighetskrav är det inte acceptabelt med mjukvarusystem som kräver upprepade uppdateringar, eftersom sådana system antagligen inte går att lita på i tillräckligt hög grad för att låta dem hantera känslig information. Istället behövs mjukvarusystem som redan från början kommer att uppföra sig på ett korrekt sätt.
Denna rapport ger en översikt över existerande forskningslitteratur inom fältet *pålitliga IT-plattformar*. Detta görs i form av en litteraturstudie av metoder och tekniker för att påvisa att mjukvara som körs på en IT-plattform är pålitlig, i betydelsen att den beter sig som den ska. En sökning gjordes i databasen Scopus. Totalt hittades 113 artiklar som var relevanta för studien och dessa kategoriserades enligt ett antal kriterier.
Studien visar att vissa kategorier har fått mer uppmärksamhet än andra och att vissa tekniker (t.ex. trusted platform module) dominerar. En annan observation är att de flesta artiklarna, särskilt de som inte är så gamla, studerar mycket små och smala problem och att det är svårt att se hur dessa lösningar på ett enkelt sätt ska kunna läggas samman till en helhet som kan anses vara pålitlig.

Nyckelord

pålitlig plattform, pålitlig mjukvara, högassuransmjukvara, litteraturstudie

Abstract

In today's very large software systems, it is common to provide software updates over time to correct errors and to improve functionality. In operations with high confidentiality and/or availability requirements, software systems that require frequent updates are not acceptable, since such systems are probably not sufficiently trusted to allow them to handle sensitive information. Instead it is necessary to have software systems that from the beginning behave properly.
This report provides an overview of existing research papers in the area of *trusted platforms*. The overview is based on a literature review of methods and techniques designed to ensure that the software running on an IT platform can be trusted, in the sense that it behaves as expected. A search was performed in the Scopus database and the articles found were categorized according to several criteria.
The study shows that certain categories have received more attention than others and that some technologies (e.g. Trusted Platform Module) dominate. Another observation is that most of the articles, especially those who are not so old, study small and narrow problems and that it is difficult to see how these solutions in a simple way can be combined to create a platform that can be considered reliable.

Keywords

trusted platform, trusted software, high assurance software, literature review

Innehåll

1 Inledning

Hårdvara har under lång tid blivit allt billigare, effektivare, snabbare och mindre. Detta gäller inte minst digitalteknik och vi är nu i en situation där många vardagsföremål innehåller stor beräkningskraft. Allt oftare ersätts specialgjord hårdvara med mjukvara, dvs. mjukvarustyrda system som körs på generella processorer. Det finns stora fördelar med detta eftersom det går att använda billiga standardkomponenter. De speciella behoven i varje enskild tillämpning tillfredsställs genom att hårdvarans beteende styrs med mjukvara. Detta möjliggörs av att hårdvara med tillräcklig snabbhet för att exekvera programmen, och minnen med tillräcklig storlek för att lagra programmen, idag är så billiga. Jämfört med hårdvarulösningar är mjukvarulösningar flexibla och lätta att utveckla.

Genomgående går utvecklingsverktygen för mjukvarusystemen mot allt högre grad av abstraktion och allt fler (hierarkiska) mjukvarunivåer. Den ökade abstraktionen syns exempelvis hos programmeringsspråken som inte längre behöver avspegla den underliggande hårdvaran, utan istället används annan mjukvara för att få hårdvaran att bete sig på det sätt som programmeringsspråket förutsätter. Det ökade antalet mjukvarunivåer syns tydligast i mer kraftfulla system, där kommunikation mellan användarprogram och hårdvara sker via drivrutiner, operativsystem och interpretatorer, eller kanske med stöd av omfattande infrastruktur i form av funktionsbibliotek och annan färdigskriven kod som användarprogram kan använda sig av.

Kombinationen av ökad abstraktion och fler mjukvarunivåer gör att även till synes enkla program genererar stora körbara filer. I fallet med interpreterande språk blir kanske inte programfilen stor, men i gengäld krävs en interpretator, som i sin tur är stor. Förutom de körbara filerna finns en infrastruktur av körbar kod i operativsystemet. Denna syns inte i den körbara filens storlek men är omfattande och behövs för att program i allmänhet ska kunna exekvera korrekt. En nackdel med utveckling i miljöer med hög abstraktionsgrad är att användarprogram använder sig av, eller är beroende av, stora mängder mjukvara för att kunna fungera. Större delen av denna mjukvara är gjord av någon annan och det finns mycket små möjligheter att veta något om vad den egentligen gör eller vilken kvalitet den har.

Den kommersiella lösningen för dessa mycket stora mjukvarusystem är att tillgängliggöra uppdateringar av mjukvaran efter hand som fel korrigeras och funktionalitet förbättras. Denna strategi har i många avseenden varit framgångsrik och ger användarna tidig tillgång till ny teknik, som dock kan behöva uppdateras några gånger innan den når sin fulla potential. Modellen har dock svagheter ur ett säkerhetsperspektiv, men i de fall användarnas sekretessbehov är begränsat kan säkerheten i allmänhet hållas på en god nivå genom ett välfungerande system för tillförlitlig hantering av data (t.ex. säkerhetskopiering). Detta förutsätter dock också att användarna kan acceptera korta funktionsbortfall vid uppgraderingar eller när mjukvara fallerar och kräver omstart eller ominstallation.

I verksamheter med höga sekretesskrav och/eller tillgänglighetskrav är situationen emellertid en annan. Då kan inte oplanerade funktionsbortfall accepteras och det räcker inte att data finns tillgänglig någonstans – den får *inte* finnas någon annanstans än på rätt ställe. Mjukvarusystem som kräver upprepade uppdateringar går antagligen inte att lita på i tillräckligt hög grad för att låta dem hantera känslig information. Istället behövs mjukvarusystem som

redan från början kommer att uppföra sig på ett korrekt sätt. För att känna tilltro måste det finnas goda skäl att tro att det är på det sättet, goda skäl som måste kunna visas upp, granskas, analyseras och dokumenteras.

Det finns flera områden med mjukvarusystem med krav på högt förtroende. Som exempel på verksamhet med höga tillgänglighetskrav kan nämnas flyget, både vad gäller system i luften och system på marken. Sekretesskrav finns på flera ställen, men närmast till hands och kanske också mest påtagligt, är behovet inom försvarssektorn där hög grad av sekretess har en lång tradition.

Frågan om hur stora IT-system kan göras pålitliga har behandlats både ingenjörsmässigt och forskningsmässigt åtminstone sedan 1970-talet och de grundläggande principer som beskrevs då är fortfarande giltiga och används i viss utsträckning i moderna operativsystem. Under 2000-talet började det också komma allmänt tillgänglig hårdvara för att skapa pålitlighet hos generella IT-system. Det mest kända exemplet på sådan hårdvara är antagligen Trusted Platform Module (TPM) [1], men det finns fler som Never Execute Bit (NX bit)[2], Intel Trusted Execution Technology (Intel TXT) [3] och AMD Virtualization (AMD-V) [4]. Dessa allmänt tillgängliga tekniker har i sin tur inspirerat till en stor mängd forskningspublikationer på området.

I den här rapporten presenteras en litteraturstudie av tekniker och metoder för att säkerställa att den mjukvara som körs i ett IT-system är pålitlig, i betydelsen att det går att verifiera att den uppför sig som den ska. Hur den *ska* uppföra sig kan vara dokumenterat i en specifikation av något slag eller helt eller delvis baseras på en allmänt spridd uppfattning om vad som är rimligt att förvänta sig. I det senare fallet är det förstås svårare, eller till och med i vissa avseenden omöjligt, att avgöra i vilken utsträckning systemet är pålitlig. En mer ingående presentation av syftet och avgränsningarna i den här rapporten finns i kapitel 3, efter kapitel 2 som introducerar nödvändig terminologi och en modell för mjukvarans livcykel. Vidare presenteras resultat i kapitel 4 och en översikt över liknande studier i kapitel 5. Vi avslutar rapporten med diskussion och slutsatser i kapitel 6.

2 Bakgrund

Detta kapitel presenterar material som behövs för att läsa resten av rapporten. Det inledande exemplet problematiserar och belyser pålitlighetsaspekterna hos en IT-produkt och kan ses som en motiverande förklaring till studiet av pålitlighetsfrågor. Presentationen av relevanta termer och de tolkningar av dessa som används i rapporten är nödvändig för att inte den stora mängden alternativa definitioner som existerar ska göra materialet svårtolkat. Livscykelmodellen är skelettet som vi hänger upp hela litteraturstudien på. Den avslutande beskrivningen av Trusted Platform Module (TPM) motiveras av att det är den enda hårdvarubaserade tekniken inom området som har en allmän spridning och därför med vid marginal är det enskilt mest frekventa objektet för studier och analys i de artiklar vi hittat i litteraturstudien.

2.1 Pålitlighetsexempel

Från [5] har vi hämtat följande exempel på hur ett till synes enkelt system egentligen är ganska komplext – i det här fallet bredbandsroutrar[1] för konsumentbruk. I dessa finns i allmänhet, förutom näthårdvaran, en generell processor som kör ett operativsystem, på vilket ett flertal servertjänster kör. Det finns till exempel nästan alltid en webbserver med vars hjälp administration och konfiguration kan göras och det är inte ovanligt att det också finns säkerhetsrelaterade funktioner. Exempel på detta är blockering av vissa externa webadresser och möjlighet att begränsa internetåtkomsten för interna datorer baserat på tidpunkt (veckodag och klockslag) och trafiktyp.

För en hemmamiljö är det troligen tillräckligt att tillverkaren försäkrar att systemet är säkert för att de flesta nyttjare ska känna sig tillräckligt trygga med att det fungerar som det ska. I ett sammanhang med höga krav på säkerhet skulle dock ett antal frågor infinna sig:

- Hur pålitliga är de som designat hård- och mjukvaran i routern?
- Har designerna gjort rätt i sin bedömning av hotbild, risknivå och säkerhetsbehov?
- Hur skickliga är utvecklarna och hur väl fungerar utvecklingsmetoden de använt för att göra säkra implementationer?
- Har tillverkaren av routern använt hård- eller mjukvara som de inte själva har utvecklat? I så fall, hur pålitlig är denna?
- Hur har överföringen av källkod till maskinkod gjorts? Kan det finnas svagheter i kompilatorn, antingen på grund av buggar eller på grund av att någon medvetet fått den att bete sig på något speciellt sätt?
- Innehåller just det här exemplaret av routern den mjukvara den ska innehålla och ingenting annat?
- Är konfigurationen av routern korrekt?
- Går det att påverka routern efter uppstart så att den beter sig felaktigt?

[1] En bredbandsrouter är oftast en ethernetswitch för ett lokalt nät, ihopkopplat med en gateway för anslutning till ett externt nät. Ofta finns ytterligare funktioner för att hantera det interna och det externa nätet.

Dessa frågor handlar i stor utsträckning om riktighet och pålitlighet. Frågorna visar att även om det finns kända säkerhetsfunktioner, som är anpassade för den existerande hotbilden, är frågorna kring riktighet och pålitlighet ett kvarstående problem som måste lösas separat.

2.2 Terminologi

Här presenteras de begrepp som används i rapporten, tillsammans med tillhörande tolkning. Terminologin inom området är rikhaltig. För den som vill få en bredare syn på denna finns i appendix A en mer omfattande lista med engelska och svenska termer, tillsammans med tillhörande definitioner från ett flertal olika källor.

Assurans

Tillit till att ett systems eller en produkts säkerhetsfunktioner uppfyller specificerade säkerhetskrav [6]. Vi använder termen assuransnivå som mått på graden av tillit vi har till systemet eller produkten.

Attestering

Att med en väldefinierad metod bevisa eller verifiera ett påstående om ett objekt. Vilken egenskap som avses måste specificeras på något sätt.

Plattform

En dator, inte nödvändigtvis i kontorsutförande.

Pålitlighet

Egenskap att riktigheten kan attesteras.

Riktighet

Egenskap eller tillstånd som innebär att information eller funktion stämmer överens med relevant specifikation och inte har ändrats, vare sig obehörigen eller av misstag.

Säkerhet

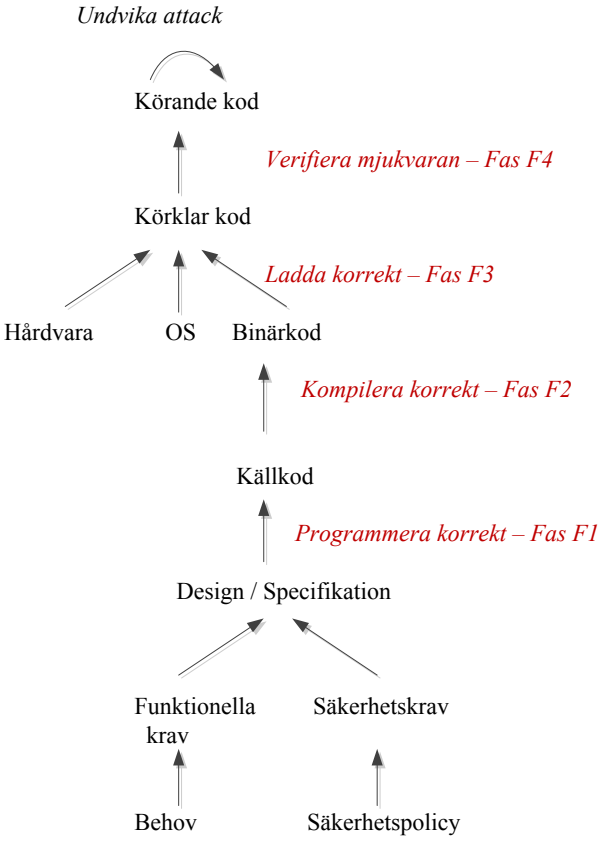
Egenskap eller tillstånd som innebär skydd mot risk för oönskad insyn, förlust eller påverkan; oftast i samband med medvetna försök att utnyttja eventuella svagheter. [6]

Vi menar att begreppet pålitlighet är starkt kopplat till vilken assuransnivå som har uppnåtts och att det går att uppnå olika assuransnivåer beroende på valet av attesteringsmetoder. Pålitlighet är önskvärd under utveckling, installation och användning av mjukvara. Det är därför intressant att identifiera vilka attesteringsmetoder som finns för olika faser i ett IT-systems livscykel och vilken assurans som kan uppnås.

2.3 Modell

För att kunna strukturera resultatet i litteraturstudien på ett relevant sätt har det behövts en kategoriindelning. Den huvudsakliga indelningen är gjord baserad på var i IT-systemets livscykel som artiklarna är applicerbara. En enkel modell bestående av fyra faser används. Modellen beskriver inte hela livscykeln hos ett IT-system, utan bara de delar som behövs i denna studie. Modellen tar sin början när det finns en specifikation för systemets beteende, och slutar när

en plattform innehåller körande kod. I Figur 2.1 nedan visas hela livscykeln hos ett IT-system och de fyra faserna F1–F4 i sin kontext.



Figur 2.1: Den modell som används i denna litteraturstudie, insatt i sin kontext.

För varje fas kan olika attesteringsmetoder användas för att försäkra sig om att specificerade mål verkligen har uppnåtts (t.ex. att binärkoden som resulterar efter kompileringen är resultatet av en korrekt kompilering). I Figur 2.1 har vi med rött formulerat en övergripande beskrivning av vad som behöver attesteras:

- Mellan specifikation och källkod (F1) måste det säkerställas att källkoden uppfyller specifikationen.
- Mellan källkod och binärkod (F2) måste det säkerställas att binärkoden är en korrekt avbildning av källkoden.
- Mellan binärkod och körklar kod (F3) måste det säkerställas att det som laddas in i IT-systemet är en oförvanskad version av binärkoden.
- Mellan körklar kod och körande kod (F4) måste det vid uppstart säkerställas att den körklara koden inte har påverkats sedan den installerades i IT-systemet.

2.4 Trusted Platform Module (TPM)

Lösningar baserade på TPM är mycket vanliga i litteraturen. Därför ges här en kort beskrivning hämtad huvudsakligen från [5].

TPM är en plattform vars utveckling drivs av Trusted computing group (TCG) som är ett initiativ startat av AMD, HP, IBM, Intel, Microsoft m.fl. Målet för TCG är att utveckla *trusted computing*. Med trusted computing menas huvudsakligen att det ska gå att verifiera att endast auktoriserad mjukvara kör i ett system, något som är besläktat med det vi har kallat en pålitlig plattform.

TPM-tekniken är relativt väl spridd. TPM-stöd finns sedan flera år hos komponenter från de flesta stora tillverkare av datordelar, från moderkort och CPU:er till bärbara och stationära datorer. Visst stöd i mjukvara finns hos både Windows och Linux och fortsatt utveckling pågår. Som exempel kan nämnas att amerikanska försvarsdepartement kräver att alla nya datorer ska innehålla ett TPM-chip [7].

TPM innehåller flera tekniker och funktioner, där de viktigaste är:

- Signeringsnyckel (engelska: endorsement key)

  Varje TPM-chip innehåller en hemlig signeringsnyckel som kan användas för att skapa digitala signaturer. Nyckeln är en 2048-bitars RSA-nyckel som skapas vid tillverkningen av chippet och kan inte ändras eller läsas i efterhand.

- Minnesskydd (engelska: memory curtaining)

  Minnesskyddet i TPM-plattformar är mycket starkare än i vanliga datorer. Delar av primärminnet kan göras helt avskilda från resten av systemet, så att varken användare eller operativsystem kan nå det. Den exakta utformningen av skyddet är implementationsberoende och kan variera mellan tillverkare. En implementation är Trusted execution environment från Intel, som kräver ett TPM-chip och särskilt hårdvarustöd i CPU och moderkort för att fungera.

- Förseglad lagring (engelska: sealed storage)

  Förseglad lagring gör det möjligt att göra information tillgänglig bara till en viss kombination av hårdvara och mjukvara. På detta sätt går det att förhindra känslig information från att bli tillgänglig för hård- eller mjukvara som inte är pålitlig ur informationsägarens perspektiv.

- Fjärrattest (engelska: remote attestation)

  Fjärrattesten gör det möjligt för TPM-chippet i en dator att på ett pålitligt sätt visa för en annan maskin, via ett nät, vilken hård- och mjukvara som används. På detta sätt får den andra maskinen ett underlag på vilket den kan basera beslut angående om den ska betrakta den första maskinen som pålitlig eller inte.

Värt att påpeka är att TPM alltid måste aktiveras för att kunna användas. Den som köper en ny dator riskerar alltså i dagsläget inte att bli påtvingad TPM-funktionerna. En risk som påtalats är annars att TPM-chippet ska användas för otillbörlig begränsning av vad användaren kan göra med sin egen information, hård- och mjukvara.

3 Litteraturgenomgång

I det här kapitlet presenteras studiens syfte samt avgränsningarna och metoden som har använts.

3.1 Syfte och avgränsningar

En litteraturgenomgång har gjorts i syfte att identifiera vilka attesteringsmetoder som finns för att verifiera att ett system har de egenskaper som efterfrågats. Vidare ska områden identifieras där det saknas forskning eller där det finns möjligheter att forska vidare. Vid systembygge ska studien kunna användas som utgångspunkt för att identifiera relevant litteratur som beskriver hur pålitlighet kan uppnås.

Alla faser i livscykeln hos en IT-produkt är viktiga avseende pålitlighet, men den här studien berör bara de fyra faser som beskrivs i modellen i avsnitt 2.3 (från specifikation till att körklar kod körs på datorn) och inte i alla avseenden. Exempelvis studeras inte hårdvarufrågor och inte heller software-engineering-frågor i fas F1. Vi har tittat på lokala attesteringar, och har därför uteslutit distansattestering från studien. Vi har inte heller tittat på körande system, utan vårt intresse slutar när systemet är körklart.

3.2 Metod

För att identifiera relevant litteratur genomfördes i april 2013 en sökning i Scopus, en databas med vetenskaplig litteratur på sammanfattnings- och referensnivå. Det sökbegrepp som användes var:

```
  TITLE-ABS-KEY(application OR software OR platform OR platforms
OR system OR systems) AND TITLE-ABS-KEY(assurance OR ‘‘integrity
measurement’’ OR attestation OR attest OR verification OR verify
OR certify OR certification OR validate OR validation OR test
OR testing OR assert OR assertion OR guarantee OR evidence OR
trojan OR backdoor)
```

Sökningen resulterade i 1051 artiklar som rapportförfattarna gick igenom för att hitta de som var relevanta för den här studien. En första genomgång, där endast titel och sammanfattning betraktades, resulterade i att 745 artiklar sorterades bort. 306 artiklar utreddes vidare och för 257 av dessa hittades artikeln i fulltext. Då det inte ansågs vara möjligt att göra en korrekt bedömning utan tillgång till hela artikeln, sorterades övriga artiklar bort.

Av de 257 artiklarna valdes 25 ut slumpmässigt och båda rapportförfattarna läste och klassificerade dessa. Tio av dessa artiklar bedömdes vara relevanta för studien. Därefter diskuterades de respektive valen och jämkades samman i syfte att ”kalibrera” bedömningarna. Övriga 232 artiklar fördelades mellan rapportförfattarna och klassificerades individuellt. Efter denna noggrannare genomgång kvarstod 113 artiklar som bedömdes vara relevanta för några av de fyra faserna i modellen, vilket var vad som användes som slutgiltigt kriterium för artiklar att inkluderas i litteraturstudien. För var och en av dessa artiklar bedömdes dessutom om den är TPM-relaterad, om eventuell teknik eller metod finns att tillgå, om den är testad och om den har en bevisad effekt. För varje artikel beskrevs också innehållet med en mening. All denna data finns i appendix B.

För att ytterligare strukturera de resulterande artiklarna delades de in i tre kategorier: *kategori A* för artiklar som föreslår ramverk eller modeller, *kategori B* för artiklar som innehåller erfarenheter, diskussioner, analyser, översikter eller liknande och *kategori C* för artiklar som innehåller metodförslag, lösningar eller implementationer.

När kategoriindelningen var färdig gjordes den analys av artiklarna, vars resultat beskrivs i kapitel 4.

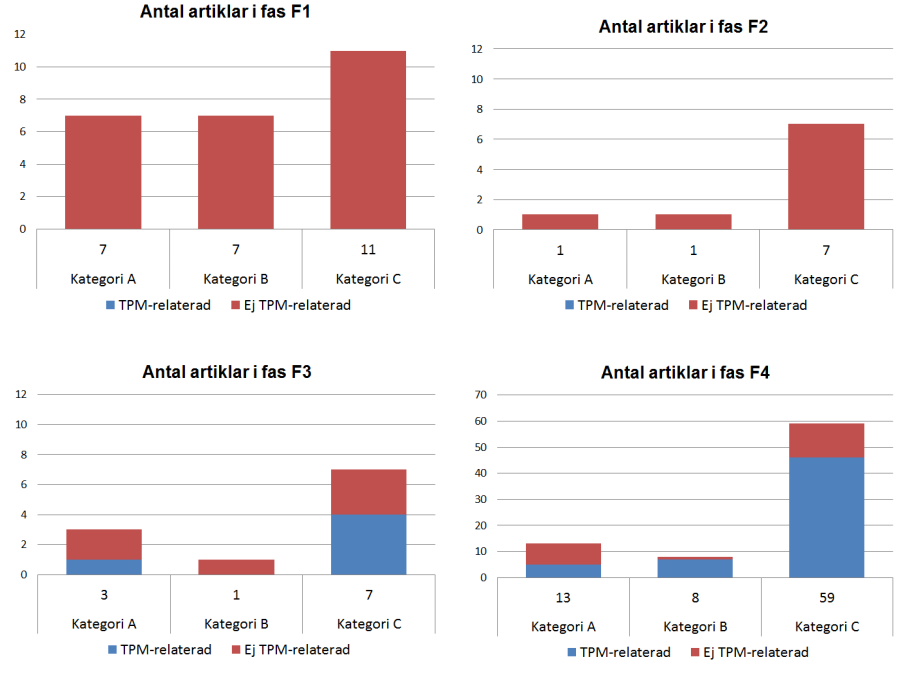
4 Resultat

I detta kapitel presenteras en översikt över de 113 artiklar som inkluderades i studien.

4.1 Fasperspektiv

Över de faser som definieras i avsnitt 2.3 fördelade sig artiklarna enligt följande, med vissa artiklar som bedömdes relevanta för mer än en fas: 25 artiklar i fas F1 – programmera korrekt, 9 artiklar i fas F2 – kompilera korrekt, 11 artiklar i fas F3 – ladda korrekt, och 80 artiklar i fas F4 – verifiera mjukvaran. I figur 4.1 visas fördelningen i faser grafiskt. Observera att skalan i fas F4 skiljer sig från den i övriga faser.

Av de 113 inkluderade artiklarna, bedömdes två vara relevanta för alla fyra faser. En av dessa presenterar ett ramverk för att bygga pålitliga system [8] och en presenterar en säkerhetsmodell för flernivå-säkerhet [9]. Båda är gamla artiklar, från 1988 respektive 1989. Utöver dessa bedömdes några artiklar som relevanta för både fas F1 och fas F2 och några som relevanta för både fas F3 och fas F4.



Figur 4.1: Antal artiklar i varje fas, indelade i de olika kategorierna A, B och C. Blå färg anger koppling till TPM och röd färg anger att sådan koppling saknas. Notera att y-axeln för fas F4 skiljer sig från de övriga.

4.1.1 Fas F1

Till fas F1 hör metoder och tekniker som används i övergången från specifikation till källkod, vilket ska göras på ett riktigt sätt. Av de 25 artiklarna som placerades i fas F1, placerades sju artiklar i kategori A. De flesta av dessa handlar om att bygga pålitliga system eller mjukvara, eller om att assurera mjukvara.

Sju artiklar placerades i kategori B. En del av dessa beskriver erfarenheter eller analyser av en specifik typ av mjukvara, som till exempel högassuransmjukvara eller säkerhetskritisk mjukvara. Vidare identifierades en diskussion om hur det kan verifieras att mjukvaran överensstämmer med specifikationen [10] och om hur formella metoder kan användas för att skapa bättre mjukvara [11]. Elva artiklar placerades i kategori C. Det finns inget tydligt gemensamt fokus bland dessa artiklar, men några exempel inkluderar bland annat en metod för att skriva återanvändbar kod inom högassuransområdet [12], automatisk kodgenerering tillsammans med automatgenererade bevis för korrektheten [13] och en metod för att beskriva krav på ett entydigt och testbart sätt [14].

4.1.2 Fas F2

Till fas F2 hör metoder och tekniker som används i övergången från källkod till binärkod, dvs. korrekt kompilering. I den här fasen placerades minst antal artiklar, endast nio, och av dessa bedömdes fem också vara F1-relevanta. En av de nio artiklarna artiklarna placerades i kategori A, en i kategori B och sju placerades i kategori C. Exempel på innehåll i denna fas är en metod för att skapa en formellt verifierad kompilator [15], en temporallogik för verifiering av adaptiv mjukvara [16] och automatisk verifiering av objektkod mot källkod för små program [17]. Av de nio artiklarna föreslår tre användning av formella metoder och två föreslår användning av en automatiserad process för kodgenerering eller verifiering.

4.1.3 Fas F3

Till fas F3 hör lösningar för att ladda korrekt. Liksom i fas F2, har väldigt få artiklar placerats i fas F3, totalt 11, varav två är de som bedömdes vara relevanta för alla fyra faser. Tre av artiklarna placerades i kategori A, en artikel placerades i kategori B och resterande 8 placerades i kategori C. Metoderna berör huvudsakligen olika förbättringar och förslag på hur integritetskontroller av binärkod ska göras på ett bra sätt. Exempel från dessa fas är ett ramverk för säker nedladdning och installation av mjukvara [18] och en lösning för säker installation över nätverk [19]. Fem av de elva artiklarna är TPM-relaterade.

4.1.4 Fas F4

Till fas F4 hör lösningar för att verifiera mjukvaran efter installation på datorn, i övergången från körklar kod till körande kod. Av de 80 artiklar som placerades i fas F4 är 58 artiklar TPM-relaterade. Tretton artiklar placerades i kategori A, åtta placerades i kategori B och 59 artiklar placerades i kategori C. I kategori A finns det till exempel olika ramverk för integritetskontroll av mjukvara. I kategori B finns det till exempel kritik och problematisering av området trusted computing [20; 21]. I kategori C finns fjorton artiklar som presenterar förslag på förbättringar, vidareutveckling eller olika varianter av TPM och trusted computing, men det finns också artiklar med lösningar för trusted computing i situationer där TPM-chip saknas och sju artiklar som presenterar lösningar där TPM används för olika säkerhetsåtgärder.

4.2 Tidsperspektiv

Utgående från hur antalet artiklar i de olika faserna varierar över tiden, tycks forskningsintresset med tiden ha gått mer och mer mot fas F4 (se tabell 4.1). Det kan också konstateras att det generellt finns färre artiklar inom faserna F2 och F3 än i de två andra faserna, men att detta huvudsakligen beror på

Tabell 4.1: Tidsindelning av fasfördelade artiklar i studien. Då vissa artiklar bedöms relevanta för flera faser är summan större än antalet artiklar.

	Antal artiklar i tidsperioden				
	F1	F2	F3	F4	Total
<1990	4	2	2	2	10
1990 – 1999	5	2	2	1	10
2000 – 2004	2	0	1	4	7
2005 – 2009	8	3	3	48	62
2010 – 2013	6	2	3	25	36
Totalt	25	9	11	80	

publikationsfördelningarna de senaste nio åren. Det kan vara så att intresset inte har varit lika stort för dessa två faser eller att det inte finns lika många intressanta forskningsfrågor, men det kan också vara så att valet av sökbegrepp missgynnar artiklar som handlar om dessa faser.

Tidiga artiklar är ofta av mer övergripande karaktär och uppmärksammar problem på ett mer generellt sätt samt diskuterar hur dessa problem kan bearbetas. Längre fram blir problemen och lösningarna mer konkreta och specifika. Från år 2005 och framåt har många TPM-relaterade artiklar hittats, och då tillhör de fas F4. Många av de TPM-relaterade artiklarna föreslår små förändringar eller tillägg som ska förbättra den lösning som TCG (Trusted Computing Group) har föreslagit. Speciellt behandlas problematiken kring statisk integritetskontroll. Vidare, av de artiklar som är i fas F4 men som inte är TPM-relaterade, så försöker en del uppnå TPM:s funkionalitet med endast mjukvara, så att lösningen ska kunna tillämpas på datorer som inte har ett TPM-chip.

5 Liknande studier

För att placera den här litteraturstudien i en kontext presenteras en kort översikt över liknande, redan existerande studier. För varje studie ges en kort sammanfattning av vad den innehåller.

Stacy Nelson, *Survey of Software Assurance Techniques for Highly Reliable Systems*, 2004 [22]

Studien går igenom olika standarder som används i olika industrier för att assurera att programvara kommer att bete sig enligt specifikation. Författarna skriver (alla referenser i citaten kommer från originaltexten):

> Software plays an increasingly crucial role in all aspects of modern life, from flight to driving to power generation to weapons to medical devices, etc. Therefore, we must be able to trust that software is reliable and will act according to intended design rather than exhibiting errant behaviors.

Software assurance defineras som:

> The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures. [IEEE 610.12] For NASA this includes the disciplines of Software Quality (functions of Software Quality Engineering, Software Quality Assurance, Software Quality Control), Software Safety, Software Reliability, Software Verification and Validation, and IV&V [1].”

Samtidigt defineras software life-cycle som:

> The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase. [IEEE 610.12]

För varje standard de presenterar reflekterar de över den pålitlighet som kan uppnås och de förbättringsmöjligheter som finns.

Goertzel et al., *Software Security Assurance: A State-of-the-Art Report (SOAR)*, 2007 [23]

Studien behandlar ämnet assurans av säker mjukvara och fokuserar på två aspekter: hur säker mjukvara kan produceras och hur det assureras att säker mjukvara har producerats. Mest tyngd ligger på första delen, hur säker mjukvara kan produceras. Tidsramen för studien är 2002–2007. Definitionen som används för assurans (engelska: software security assurance) är:

> The basis for gaining justifiable confidence that software will consistently exhibit all properties required to ensure that the software, in operation, will continue to operate dependably despite the presence

[1] IV&V - Independent Verification and Validation

> of sponsored (intentional) faults. In practical terms, such software must be able to resist most attacks, tolerate as many as possible of those attacks it cannot resist, and contain the damage and recover to a normal level of operation as soon as possible after any attacks it is unable to resist or tolerate.

Begreppet säker mjukvara (engelska: secure software) defineras som: "Secure software cannot be intentionally subverted or forced to fail. It is, in short, software that remains correct and predictable in spite of intentional efforts to compromise that dependability."

De har kommit fram till att (åtminstone) 2007 fanns oro över att mycket mjukvara outsourcats och utvecklas av utländska aktorer och att utländska programmerare skulle kunna lägga skadlig kod i mjukvaran. Den bästa lösningen skulle varit att tänka efter angående hur inköp görs och att tillhandahålla riktlinjer för att hjälpa inköparen att bygga viss tillit till mjukvaruutvecklaren. En annan oro var att skadlig kod skulle läggas i mjukvaran innan distribution. Det fanns också ett behov av att bättre definiera vad skadlig kod betyder. Lösningar inkluderar riktlinjer för att hantera skadlig kod innan installation. En annan slutsats var att formella metoder nyligen börjat användas för att assurera de säkerhetsegenskaper som mjukvarustyrda system behöver. Värt att notera är att studien är gjord ur ett amerikanskt perspektiv.

Bryan Parno, Jonathan McCune och Adrian Perrig, *Bootstrapping Trust in Commodity Computers*, 2010 [24]

Studien handlar om hur tillit till uppstartsekvensen i datorer kan byggas. Författarna tittar på forskning kring hur en dators tillstånd på ett säkert sätt kan avgöras och hur den informationen kan användas för att förbättra säkerheten på den lokala datorn eller för att kommunicera en fjärrdators tillstånd. De tittar både efter lösningar för att verifiera kodidentitet och dynamiska egenskaper hos mjukvara som körs. De tittar också på olika hårdvarulösningar för att bygga ett skyddat lagringsutrymme, bland annat med hjälp av TPM. Studien inkluderar en kort diskussion om hur både hårdvaran och de protokoll som används kan valideras för att bygga tillit och beskriver olika applikationer som kan dra nytta av att kunna bygga tillit till en dator.

Teodor Sommestad och Jonas Hallberg, *An overview of trust attestation methods*, 2011 [25]

Studien beskriver en systematisk litteraturstudie (systematic literature review) för att besvara frågan "Which attestation methods can be used during a computer system's lifecycle in order to make the computer system's owner trust the system?" Definitionen av attestering (engelska: attestation) som används är "...the activity of making a claim to an appraiser about properties of a target by supplying evidence which supports that claim". Dokumentet använder en enkel livscykelmodell för datorsystem, bestående av utveckling, urval, installation och användning. Studien tar med både hård- och mjukvara, men inte rena säkerhetstekniker som inte skapar någon attest.

6 Diskussion och slutsater

De studier som presenteras i kapitel 5 ska ses som en bakgrund mot vilken denna rapports litteraturstudie kan betraktas. Studierna täcker en del eller delar av vad som ingår i den här litteraturstudien, eller är på något sätt relaterade till det som studeras. Nelson [22] tittar på standarder som används i olika industrier där kod med hög pålitlighet behövs, såsom flygindustrin och kärnkraftsindustrin, men tittar inte alls på forskningsfronten. Parno et al. [24] tittar på hur en dators aktuella status kan fångas, för att kunna avgöra om det går att lita på den eller inte. De tittar inte däremot på hur mjukvaran ska utvecklas så att det går att lita på den. Goertzel et al. [23] tittar på hur säker mjukvara kan utvecklas och attesteras. Deras betoning ligger dock på utveckling av sådan mjukvara, där möjliga sårbarheter och möjliga åtgärder beskrivs. Sommestad och Hallberg [25] har gjort den studie som ligger närmast den här studien. De har också använt en livscykelmodell som liknar den som används i den här studien, men har valt att även ta med hårdvaruattesteringar.

Av de 113 artiklar som behandlats i denna studie är nästan hälften TPM-relaterade. Även om TPM-relaterade lösningar är användbara och mycket forskning finns kring dem, så måste det kommas ihåg att de inte löser hela problemet med mjukvara som går att lita på. TPM kan bara verifiera och säkerställa att koden inte har ändrats sedan kontrollsumman beräknades. Den kan inte alls verifiera vilka egenskaper koden ursprungligen hade. Det behövs fortfarande tekniker från faserna F1, F2 och F3 för att säkerställa att mjukvaran gör det som den förväntas göra.

Vi har hittat flera metoder och diskussioner om hur formella metoder kan användas i olika delar i utvecklingen och hur olika delar kan automatiseras vid utvecklingen av mjukvara. Det senare är intressant eftersom det eliminerar den mänskliga faktorn, men förutsätter att implementationen av mjukvaran som automatisk genererar kod är korrekt. Problemet med den mänskliga faktorn förflyttas alltså till ett enskilt stycke kod som kan genomgå särskild kontroll. Problemet med den mänskliga faktorn finns även hos manuella formella metoder, där misstag kan begås i bevisföringar.

De flesta artiklar, särskilt de som inte är så gamla, studerar mycket små och smala problem. Det är svårt att se hur de lösningar som presenteras i sådana artiklar på ett enkelt sätt ska kunna läggas samman till en helhet som kan anses vara pålitlig. Det finns visserligen artiklar som behandlar ramverk och arkitektur men dessa är i sin tur jämförelsevis abstrakta och vaga, och erbjuder inte något lätt sätt att foga in dellösningar av det slag som de specifika artiklarna erbjuder. Det synes alltså finnas ett glapp mellan de lösningar som finns på detaljnivå och de som finns på övergripande nivå. Detta glapp kan tänkas ha olika orsaker. Det kan röra sig om att den databassökning som ligger till grund för litteraturstudien saknar söktermer som ger träffar på artiklar som kan överbrygga glappet. En annan orsak kan vara att det ligger i den moderna forskningens natur att angripa delproblem av begränsad storlek, oavsett om dessa är på detaljnivå eller på övergripande nivå. Om så är fallet är det snarare att betrakta som en ingenjörsmässig fråga att syntetisera de olika teknikerna och fylla igen de glapp som uppstår, vare sig glappen är ”på bredden” eller ”på höjden”. De ingenjörsmässiga lösningarna finns då att studera i form av existerande system, standarder och/eller metoder. Exempelvis skulle Common Criteria kunna ses som en ingenjörsmässig implementation av metoder och tekniker för pålitlighet, där ”ingenjörsmässig” inte ska tolkas som något negativt

utan snarare som att de som ligger bakom har varit tvungna att ta sig an hela problemet och skapa en metod som fungerar, utan att kunna väja för områden som är svåra eller bara delvis utforskade.

A Terminologiöversikt

I detta appendix presenteras en översikt över ett antal för området relevanta termer och en beskrivning av deras definitioner i olika sammanhang. Endast termer på engelska och svenska finns med. Den svenska terminologin är inte lika omfattande som den engelska. Detta beror på att litteraturen inom området huvudsakligen är författad på engelska, och att även i de fall den är skriven på svenska så används ändå ofta engelska termer.

I vissa fall har likartade termer från olika dokument sammanförts under en gemensam rubrik. I de fall den definierade termen avviker från termen i rubriken så anges inom parantes den exakta termen från dokumentet.

A.1 Engelsk terminologi

Assurance

- A measure of confidence that the security features and architecture of an automated information system accurately mediate and enforce the security policy. [26]
- Grounds for confidence that a TOE meets the SFRs [27] (TOE – Target of Evaluation, SFR – Security Functional Requirement).
- Grounds for justified confidence that a claim has been or will be achieved. [28]
- (security assurance) Grounds for justified confidence that a claim about meeting security objectives has been or will be achieved. [28]
- Our estimate of the likelihood that a system will not fail in some particular way. [29]

Attestation

- Issue of a statement, based on a decision following review, that fulfillment of specified requirements has been demonstrated. (The resulting statement, referred to in this International Standard as a "statement of conformity", conveys the assurance that the specified requirements have been fulfilled. Such an assurance does not, of itself, afford contractual or other legal guarantees.) [30]
- The process of vouching for the accuracy of information. External entities can attest to shielded locations, protected capabilities, and Roots of Trust. A platform can attest to its description of platform characteristics that affect the integrity (trustworthiness) of a platform. Both forms of attestation require reliable evidence of the attesting entity. [31]
- The means by which a trusted computer assures a remote computer of its trustworthy status. [32]
- The activity of making a claim about properties of a target by supplying evidence to an appraiser. [33]

Evaluation

- (security evaluation) An evaluation done to assess the degree of trust that can be placed in systems for the secure handling of sensitive information.

One type, a product evaluation, is an evaluation performed on the hardware and software features and assurances of a computer product from a perspective that excludes the application environment. The other type, a system evaluation, is done for the purpose of assessing a system's security safeguards with respect to a specific operational mission and is a major step in the certification and accreditation process. [26]

- Assessment of a PP, an ST or a TOE, against defined criteria. [27] (PP – Protection Profile, ST – Security Target, TOE – Target of Evaluation)

- Systematic determination of the extent to which an entity meets its specified criteria. [28]

- The process of assembling evidence that a system meets, or fails to meet, a prescribed assurance target. [29]

Security

- (secure state) A condition in which no subject can access any object in an unauthorized manner. [26]

- (secure state) State in which the TSF data are consistent and the TSF continues correct enforcement of the SFRs. [27] (TSF – TOE Security Functionality, SFR – Security Functional Requirement, TOE – Target of Evaluation

- Property of a system by which confidentiality, integrity, availability, accountability, authenticity, and reliability are achieved. [28]

Security policy

- The set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information. [26]

- (organisational security policy) Set of security rules, procedures, or guidelines for an organisation. [27]

- (security function policy) Set of rules describing specific security behaviour enforced by the TSF and expressible as a set of SFRs. [27] (TSF – TOE Security Functionality, SFR – Security Functional Requirement)

Trust

- The expectation that a device will behave in a particular manner for a specific purpose. [31]

- The accepted dependence of a component or system, on a set of properties of another component or system. [34]

- Firm belief in the reliability, truth, or ability of someone or something. [35]

Trusted computing

(trusted computer system) A system that employs sufficient hardware and software assurance measures to allow its use for simultaneous processing of a range of sensitive or classified information. [26]

Trusted computing base (TCB)

The totality of protection mechanisms within a computer system, including hardware, firmware, and software, the combination of which is responsible for enforcing a security policy. A TCB consists of one or more components that together enforce a unified security policy over a product or system. The ability of a TCB to enforce correctly a unified security policy depends solely on the mechanisms within the TCB and on the correct input by system administrative personnel of parameters (e.g., a user's clearance level) related to the security policy. [26]

Trusted platform

- (Trusted Computing Platform) A computing platform that can be trusted to report its properties. [31]

- (Trusted system) A system that is relied upon to a specified extent to enforce a specified security policy. As such, a trusted system is one whose failure may break a specified security policy. [36]

Trustworthiness

- The measure in which the trust component or system meets the set of properties of the other component or system (see definition of trust). [34]

- Able to be relied on as honest or truthful. [35]

Verification

Confirmation, through the provision of objective evidence, that specified requirements have been fulfilled. [37]

A.2 Svensk terminologi

Assurans

- Tillit till att ett systems eller en produkts säkerhetsfunktioner uppfyller specificerade säkerhetskrav. (engelska – assurance) [6]

- Innebär att kunna påvisa att det går att hysa tillit/ha förtroende för att det som påstås och att detta verkligen infrias. [38]

Attestera

Skriftligt intyga riktighet av dokument eller dylikt. [39]

Evaluering av IT-säkerhet

Utvärdering av en skyddsprofil, produktsäkerhetsdeklaration eller ett evalueringsobjekt gentemot definierade säkerhetskriterier. (engelska – evaluation) [6]

Korrekthet

Vid realisering av säkerhetsfunktion, egenskap att en komponents egenskaper, och dess beskrivning i olika abstraktionsnivåer/beskrivningsnivåes, överensstämmer med specificerade säkerhetskrav. (engelska – correctness) [6]

Pålitlig

Som man (alltid) kan lita på. [39] (att lita på – ha fullt förtroende för någon eller något)

Pålitlig platform

En plattform på vilken det är möjligt att verifiera riktigheten i mjukvaran som körs. [5]

Riktighet

Skyddmål att information inte förändrats, vare sig obehörigen, av misstag eller på grund av funktionsstörning. (engelska – data integrity) [6]

Säkerhet

Egenskap eller tillstånd som innebär skydd mot risk för oönskad insyn, förlust eller påverkan; oftast i samband med medvetna försök att utnyttja eventuella svagheter. [6]

Säkerhetsbetrodd

Vid systemkonstruktion, uppfattning om att man i säkerhetshänseende kan lita på en funktion eller komponent. (engelska – trusted) [6]

Verifiering

- Fastställande av riktigheter av något, med avseende på specifikation. (engelska – verification) [6]
- (verifikation) Bestyrkande av någots riktighet [39]

B Litteraturstudiens artiklar

Titel	Författare	Publikation	År	Fas	TPM-relaterad	Finns att tillgå	Testad	Bevisad effekt	Vad gör tekniken eller vad skyddar tekniken/metoden mot?
Element-level classification with A1 assurance	T. Lunt, D. Denning, R. Schell, M. Heckman, W. Shockley	Computers and Security	1988	F1, F2, F3, F4	Nej	Nej	Nej	Nej	Idé om att bygga en SCSEC A1-klassad databas, baserat på C2-komponenter och en A1-referensmonitor.
An introduction to the SMITE approach to secure computing	C. Harold	Computers and Security	1989	F1, F2, F3, F4	Nej	Nej	Ja	Nej	Ramverk för att bygga pålitliga system.
Research directions for automated software verification: Using trusted hardware	P. Devanby, S. Stubblebine	IEEE International Automatec Software Engineering Conferense	1997	F3, F4	Nej	Nej	Nej	Nej	Idé om en tamper-proof hårdvara som kan automatiskt verifiera och signera kod hos producenten.
Conqueror: Tamper-Proof Code Execution on Legacy Systems	Martignoni, L., Paleari, R., & Bruschi, D.	DIMVA	2010	F3, F4	Ja	Nej	Ja	Nej	software-based load-time attestation and runtime integrity on legacy systems
Research on management scheme of trusted application software	Huang, H., Wang, C.-H., & Wang, B.	International Conference on Network Computing and Information Security	2001	F3, F4	Nej	Nej	Nej	Nej	management scheme for trusted application software to verify that the software has not been modified before installing and while running....
A novel protocol for software authentication	Rongyu He, Zheng Qin, Shaojie Wu	Information Technology Journal	2010	F4	Nej	–	Nej	Nej	Ett protokoll för att verifiera mjukvaruintegritet vid boot, särskilt för mobila enheter.
A secure DVB set-top box via trusting computing technologies	Onur Aciiçmez, Jean-Pieree Seifert, Xinwen Zhang	IEEE Consumer Communications and Networking Conference	2009	F4	Ja	Nej	Nej	Nej	Systemlösning för en TV-box, baserad på TPM och virtualisering
Challenges for trusted computing	Shane Balfe, Eimear Gallery, Chris J. Mitchell, Kenneth G. Paterson	IEEE Security & Privacy	2008	F4	Ja	–	–	–	Problematisering av området Trusted Computing. Ganska TPM-tungt.
Property-based attestation for computing platforms: Caring about properties, not mechanisms	A. Sadehgi, C. Stüble	New Security Paradigms Workshop	2005	F4	Ja	Nej	Nej	Nej	Kritik av TCG:s användingsmodell för TPM och förslag på hur man kan göra istället.
Trusted computing and open source	D. Safford, M. Zohar	Information Security Technical Report	2005	F4	Ja	Ja	Ja	Nej	Översikt över TPM-stöd inom open-source-världen, mest Linux.
A protocol for property-based attestation	L. Chen, R. Landfermann, H. Löhr et al	Workshop on Scalable Trusted Computing	2006	F4	Ja	Nej	Nej	Nej	Ett sätt att använda TPM för property-based attestation istället för binary-based.

PRIMA: Policy-Reduced Integrity Measurement Architecture	T. Jaeger, R. Sailer, U. Shankar	Symposium on Access Control Models and Technologies	2006	F4	Nej	Nej	Ja	Nej	Information flow integrity, istället för load-time binary integrity. Implementerad och testad i SElinux.
Trusted Computing enabled access control for virtual organizations	J. Zhan, H. Zhang	International Conference On Computational Intelligence and Security Workshops	2007	F4	Ja	Nej	Nej	Nej	Verifierar att instanser i en grid lever upp till access-control policies
A new approach for secure and portable OS	H. Ghaleh, M. Doustari	International Conference on Emerging Security Information, Systems and Technologies	2008	F4	Nej	Nej	Ja	Nej	En beskrivning av hopplockade mjukvarudelar för att garantera integriteten hos mjukvaran i ett system, dessutom en jämförelse med liknande system.
A security bootstrap and measurements	F. Zhang, G. Wu, Z. Fan, L. Zhao	International Conference on Wireless Communications, Networking and Mobile Computing	2008	F4	Ja	Nej	Ja	Nej	En modifierad säker TCG-bootsekvens, som de visar vara säker och samtidigt mer flexibel.
Policy enforcement and compliance proofs for Xen virtual machines	B. Jansen, H. Ramasamy, M. Schunter	International Conference on Virtual Execution Environments	2008	F4	Ja	Nej	Ja	Nej	En utökning av TPM:s integritetskoll till policynivå, testad i Xen.
Program security inspection: Model and implementation	Z. Chen, X. Wu, W. Tang	International Conference on Wireless Communications, Networking and Mobile Computing	2008	F4	Nej	Nej	Ja	Nej	Add-on till Win-XP på låg nivå som integritetskollar körbara filer utan att ändra win-binärer.
TOCTOU, traps, and trusted computing	S. Bratus, N. D'Cunha, E. Sparks, S. Smith	Trusted Computing - Challenges and Applications	2008	F4	Ja	Nej	Ja	Nej	Visar på potentiella svagheter i integritetskollen hos TPM, och har förslag på lösning som testatd i mjukvara i Xen.
Trusted boot and platform trust services on 1CD Linux	K. Suzaki, K. Irijima, T. Yagi, A. Nguyen	Trusted Infrastructure Technologies Conference	2008	F4	Ja	Nej	Ja	Nej	De har byggt ihop Knoppix på en CD med TPM integritetskontroll.
TVDc: Managing security in the trusted virtual datacenter	S. Berger, R. Cáceres, D. Pendarakis et al	Operating Systems Review	2008	F4	Ja	Nej	Ja	Nej	En genomgång av IBM:s initiativ Trusted Virtual Datacenter, fokuserat på säkra virtuella datacentra
A novel server-based application execution architecture	C. Chen, K. Wang, S. Liao. Q. Zhan, Y. Dai	IEEE International Conference on Computational Science and Engineering	2009	F4	Ja	Nej	Ja	Nej	Någon variant av TPM-integritetskontroll. Vet inte riktigt vad som är nytt.
A practical property-based bootstrap architecture	R. Korthaus, A. Sadeghi, C. Stüble, J. Zhan	Conference on Computer and Communications Security	2009	F4	Nej	Nej	Ja	Nej	En variant av property-based secure boot, som ska vara bättre"än tidigare förslag, särskilt mot version roll-back-problematik.
HIMA: A hypervisor-based integrity measurement agent	A. Azab, P. Ning, E. Sezer, X. Zhang	Annual Computer Security Applications Conference	2009	F4	Nej	Nej	Ja	Nej	Hypervisor som kollar integriteten hos gästsystemen vid varje kritisk händelse och dessutom skyddar gästernas minne.
Research and implement of secure bootstrap for virtual machine based on trusted computing platform	Z. Zhu, M. Xu, H. Zhang	IEEE International Symposium on Dependable, Autonomic and Secure Computing	2009	F4	Ja	Nej	Ja	Nej	Design och implementation av en säker bootstrap-sekvens.
Research and realization of trusted computing platform based on EFI	F. Weiwei, Z. Changsheng, L. Yahui, Z. Liang	International Conference on Management and Service Science	2009	F4	Ja	Nej	Nej	Nej	Ersätter BIOS med en nyare variant, som integritetstestas med TPM vid uppstart.
Research on distributed and dynamic trust transfer and measurement	L. Liu, J. Peng	International Conference on Networks Security, Wireless Communications and Trusted Computing	2009	F4	Ja	Ja	Ja	Nej	En distribuerad vidareutveckling av TPM

Trusted virtual platforms: A key enabler for converged client devices	C. Dalton, D. Plaquin, W. Weidner et al	Operating Systems Review	2009	F4	Ja	Nej	Nej	Nej	Design för att kombinera Xen och TPM för att skapa trusted virtual platforms.
Integrity measurement model based on trusted virtual platform	G. Qiu, Y. Wang, L. Zhou	International Conference on Genetic and Evolutionary Computing	2010	F4	Ja	Nej	Nej	Nej	TPM och virtualisering i kombination för att skapa trusted execution environment.
Protocol for dynamic component-property attestation in trusted computing	J. Yan, X. Peng	International Conference on Networks Security, Wireless Communications and Trusted Computing	2010	F4	Ja	Nej	Ja	Nej	Utökning av property-based attestation.
Research and design of dynamic integrity measurement in trusted computing	C. Xin, S. Si	International Conference on Educational and Information Technology	2010	F4	Nej	Nej	Nej	Nej	En modell för dynamisk integritetskontroll byggd på Xen.
Scalable integrity monitoring in virtualized environments	K. Goldman, R. Sailer, D. Pendarakis, D. Srinivasan	Conference on Computer and Communications Security	2010	F4	Nej	Nej	Nej	Nej	En arkitektur för effektivare integritetsattestering för virtuella maskiner i stora datacenter
Seal-based secure boot scheme for trusted computing platform	C. Song, W. Peng, Y. Xin et al	Journal of China Universities of Posts and Telecommunications	2010	F4	Ja	Nej	Nej	Nej	En variant av secure boot.
Trusted virtual containers on demand	K. Bailey, S. Smith	Conference on Computer and Communications Security	2010	F4	Ja	Nej	Ja	Nej	OpenSolaris med TPM.
Attestation with trusted configuration machine	M. Lucyantie, H. Habibah, M. Mohd Anuar, A. Norazah	Conference on Computer Applications and Industrial Electronics	2011	F4	Ja	Nej	Nej	Nej	En variant av remote attestation.
TrustVP: Construction and evolution of trusted chain on virtualization computing platform	D. Xue, X. Wu, Y. Gao et al	International Conference on Computational Intelligence and Security	2012	F4	Ja	Nej	Ja	Nej	Arkitektur och metod för att skapa en trusted chain i virtuella miljöer.
A Demonstrative Ad Hoc Attestation System	Endre Bangerter, Maksim Djackov, and Ahmad-Reza Sadeghi	Lecture Notes in Computer Science	2008	F4	Ja	Nej	Ja	Nej	använder en security token (external device) för att attestera en platförm
A Model-driven Framework for Trusted Computing based Systems	Alam, M., Seifert, J.-P., & Zhang, X.	IEEE International Enterprise Distributed Object Computing Workshop	2007	F4	Ja	Nej	Ja	Nej	a model driven framework for rendering TC-related requirements at a higher level of abstraction, integration of TC-related requirements into distributed appliction development
A Multi-layered Approach to Security in High Assurance Systems	Alves-Foss, J., Taylor, C., & Oman, P.	Hawaii International Conference on System Sciences	2004	F4	Nej	Nej	–	–	multi-layer architecture for high assurance systems for embedded real time systems
A new approach to protect the OS from off-line attacks using the smart card	Ghaleh, H. R., & Norouzi, S.	3rd International Conference on Emerging Security Information, Systems and Technologies	2009	F4	Ja	Nej	Ja	Nej	proposes the use of a removable trsuted storage on devices that have no TPM
acTvSM: A Dynamic Virtualization Platform for Enforcement of Application Integrity	Ronald Toegl, Martin Pirker, and Michael Gissing	Lecture Notes in Computer Science	2011	F4	Ja	Nej	Ja	Nej	an architecture that provides integrity guarantees to the applications and services hosted on a platform
An Efficient Security Architecture for Trusted Computing	Yin Zhixi	2nd IEEE International Conference on Computer Science and Information Technology	2009	F4	Ja	Nej	Nej	Nej	trusted virtual machine monitor for TPM - en sätt att använda TPM på

An Improved Sealing Scheme for Trusted Storage	Chi YaPing , Ju Lei, Shen XiaoDong, Fang Yong	International Conference on Computational Intelligence and Software Engineering	2009	F4	Ja	Nej	Nej	Nej	improved sealing for trusted storage by introduction of root of trust for reporting and stored measurement log
An Integrity Batch Report Scheme Based on the Waiting Stack	Chang, C. et al.	Information Technology Journal	2010	F4	Ja	Nej	Ja	Nej	improve the TPM signature in integrity reporting by using integrity batch report
ARMor: Fully Verified Software Fault Isolation	Zhao, L. et al.	9th ACM International Conference on Embedded Software	2011	F4	Ja	Nej	Nej	Ja	a sandboxing system for ARM binaries based on software fault isolation to protect critical components from other less-trusted components.
BBACIMA: A Trustworthy Integrity Measurement Architecture through Behavior-Based TPM Access Control	Yu A., Feng D.	Wuhan University Journal of Natural Sciences	2008	F4	Ja	Nej	Nej	Nej	behavior-based access controll for the TPM
Behavior-based Attestation of Policy Enforcement among Trusted Virtual Domains	Yu, R.-W. et al.	Journal of Networks	2010	F4	Ja	Nej	Ja	Nej	behaviour-based attestation of policy enforcement for workflow protection, verification when security policies of two individual virtual domains are inconsistent
Concerning about trust of platform hardware	Fan, Z.a, Guoqing, W., Min, J., Xiaoli, L.	Pacific-Asia Workshop on Computational Intelligence and Industrial Application	2008	F4	Ja	Nej	Nej	Nej	hardware integrity check to avoid hardware based attacks like using a malicious processor to attack a system
Design and Implementation of Portable TPM Device Driver based on Extensible Firmware Interface	Peng, S., & Han, Z.	1st International Conference on Multimedia Information Networking and Security,	2009	F4	Ja	Nej	Ja	Nej	portable TPM for legacy systems and the driver for the portable TPM
Enforcing Executing-Implies-Verified with the Integrity-Aware Processor	Michael LeMay and Carl A. Gunter	TRUST	2011	F4	Ja	Nej	Ja	Nej	verifies code that executes on a target system against a network-hosted whitelist
Establishing and Sustaining System Integrity via Root of Trust Installation	St.Clair, L., Schiffman, J., Jaeger, T., & McDaniel, P.	Annual Computer Security Applications Conference	2007	F4	Ja	Nej	Ja	Nej	build and verify an high integrty system based on a root of trust instalation
Externally verifiable code execution	Seshadri, A., Luk, M., Perrig, A., Van Doorn, L., & Khosla, P.	Communications of the ACM	2006	F4	Nej	Nej	Nej	Nej	externally verifiable code execution - kod laddas och exeviteras korrekt, utan påverkan från mallware
Formal Analysis of Trusted Computing: One Case Study	HongWei Zhou, JinHui Yuan	3rd International Conference on Communications and Mobile Computing	2011	F4	Ja	Nej	Nej	Ja	formal analyses of trusted computing - extends the LS2 logic to include the isolation which is provided by visualiztion
Improving the Scalability of Platform Attestation	Frederic Stumpf, Andreas Fu, Stefan Katzenbeisser, Claudia Eckert	ACM Conference on Computer and Communications Security	2008	F4	Ja	Nej	Nej	Nej	proposes three protocols to improve the scalability of platform attestation most applicable in client server architectures where the server is doing the attestation for all clients
Protected JTAG	Buskey, R. F., & Frosik, B. B.	International Conference on Parallel Processing Workshops	2006	F4	Ja	Nej	Nej	Nej	protected jtag for debugging purposes after the user has received the device
Provenance-Based Model for Verifying Trust-Properties	Cornelius Namiluko and Andrew Martin	TRUST 2012	2012	F4	Ja	Nej	Ja	Nej	proveniance -based model for establishing trust, complements TCG integrity schema
Reducing TCB size by using untrusted components - Small kernels versus virtual-machine monitors	Hohmuth, M., Peter, M., Härtig, H., & Shapiro, J. S.	11th Workshop on ACM SIGOPS European Workshop	2004	F4	Ja	Nej	Ja	Nej	reduce the TCB by using VMMs with features for secure messaging and memory sharing
Secure In-VM monitoring using hardware virtualization	Sharif, M. I., Lee, W., Cui, W., & Lanzi, A.	ACM Conference on Computer and Communications Security	2009	F4	Nej	Nej	Ja	Nej	a VMM based solution to protect against kernel-level attacks...

Security Architecture of Trusted Virtual Machine Monitor for Trusted Computing	Huang Q., Shen C., Fang Y.	Wuhan University Journal of Natural Sciences	2007	F4	Ja	Nej	Nej	Nej	propose the use of VMM to implement TCG specifications
Software integrity protection using timed executable agents	Garay, J. A., & Huelsbergen, L.	ACM Symposium on Information, Computer and Communications Security	2006	F4	Nej	Nej	Nej	Nej	a software-based attestation scheme
SPEE: A Secure Program Execution Environment tool using code integrity checking	Gelbart, O., Narahari, B., & Simha, R	Journal of High Speed Networks	2006	F4	Nej	Nej	Ja	Nej	software based tool for code verification (executables)
Terra: A virtual machine-based platform for trusted computing	Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., & Boneh, D.	Operating Systems Review (ACM)	2003	F4	Nej	Nej	Ja	Nej	an architecture for trusted computing based on a tamper-resistent hardware
Trust[ed I in] computing, signed code and the heat death of the internet	Poritz, J. A.	ACM Symposium on Applied Computing	2006	F4	Ja	Nej	Nej	Nej	discusses the weakeesees of remote attestation in TC, and proposes proberty based attestation instead
Trusted integrity measurement and reporting for virtualized platforms	Cabuk, S., Chen, L., Plaquin, D., & Ryan, M.	INTRUST 2009	2010	F4	Ja	Nej	Ja	Nej	an integrity management solution for TC based on a small software root of trust
Trusting your computer to be trusted	Mason, S	Computer Fraud and Security	2005	F4	Ja	–	–	–	describes TCG and TC
Using TPM to improve boot security at BIOS layer	Lin, K.-J., & Wang, C.-Y.	IEEE International Conference on Consumer Electronics	2012	F4	Ja	Nej	Nej	Nej	a solution for secure boot based on TPM
Cerberus: A novel hypervisor to provide trusted and isolated code execution	Chen Wen-Zhi, Zhang Zhi-Peng, Yang Jian-Hua, He Qin-Ming	International Conference of Informations Science and Management Engineering	2010	F4	Ja	Nej	Ja	Nej	De har implementerat en mycket liten hypervisor för x86 som tillåter en delad skärm. TPM används för att attestera hypervisorn.
Modelling dynamic trust with property based attestation in trusted platforms	Arthi Nagarajan, Vijay Varadharajan	Data and Applications Security and Privacy	2010	F4	Nej	Nej	Nej	Nej	Istället för att bara titta på binärerna för en mjukvara så blandas även rykte och tidigare resultat in i en sammanvägd bedömning av pålitligheten hos en plattform.
TIVA: Trusted integrity verification architecture	M. Gomathisankaran, A. Tyagi	Digital Rights Management, Technologies, Issues, Challenges and Systems	2006	F4	Nej	Nej	Nej	Nej	Binär attestation utan att verifieraren behöver känna den verkliga binären. Tveksamt ang funktionen...
A high efficiency protocol for reporting integrity measurements	C. Chaowen, H. Rongyu, X. Hui, X. Guoyu	International Conference on Intelligent Systems Design and Applications	2008	F4	Ja	Nej	Nej	Nej	A new approach for secure and portable OS
Design and implementation of an integrity measurement system based on windows trusted computing platform	Yanga, H. Zhang, L. Wan, B. Zou	International Conference for Young Computer Scientists	2008	F4	Ja	Nej	Ja	Nej	Integritetskoll vid boot under windows, kompletterad med någon sorts integritetsmetod/modell under körning.
Integrity measurement enhanced security for mobile agent based on trusted computing platform	W. Xiaoping, Z. Huanguo, S. Zhidon	International Conference on Wireless Communications, Networking and Mobile Computing	2008	F4	Ja	Nej	Nej	Nej	TPM-lösning för integritetskoll för mobila agenter.
TPM meets DRE: Reducing the trust base for electronic voting using trusted platform modules	R. Fink, A. Sherman, R. Carback	IEEE Transactions on Information Forensics and Security	2009	F4	Ja	Nej	Nej	Nej	TPM för att säkra röstningsmaskiner.

A trusted integrity measurement architecture for securing enterprise network	T. Liu, P. Agrawal	IEEE International Conference on Trust, Security and Privacy in Computing and Communications	2011	F4	Ja	Nej	Nej	Nej	TPM-lösning för att visa sin integritiet för att få tillgång till ett nät.
Security, Trust and Privacy (STP) framework for federated single sign-on environment	Z. Khattak, S. Sulaiman, J.-L. Manan	International Conference on Information Technology and Multimedia: "Ubiquitous ICT for Sustainable and Green Living"	2011	F4	Ja	Nej	Ja	Nej	System för mutual attestation mellan remote-maskiner, samt metod för att verifiera user name.
A New Efficient Property-based Attestation Protocol Based on Elliptic Curves	Xiaobo Chu, Qin Yu	11th IEEE Int. Conference on Trust, Security and Privacy in Computing and Communications	2012	F4	Ja	Nej	Nej	Ja	property-based remote atestation for TC
An Implementation of a Trusted and Secure DRM Architecture	Víctor Torres, Jaime Delgado, and Silvia Llorente	Lecture Notes in Computer Science	2006	F4	Nej	Nej	Ja	Nej	architecture for Digita Right Management
Delivering Secure Applications on Commercial Mobile Devices: The Case for Bare Metal Hypervisors	Gudeth, K., Pirretti, M., Hoeper, K., & Buskey, R.	ACM Conference on Computer and Communications Security	2011	F4	Ja	Nej	Nej	Nej	an architecture that uses bare meral virtualization to provide integrity for code on mobile devices
Secure Boot Revisited	Kurt Dietrich, JohannesWinter	9th International Conference for Young Computer Scientists	2008	F4	Ja	Nej	Ja	Nej	secure boot for mobile devices
Tagging the turtle: Local attestation for kiosk computing	Toegl, R.	ISA 2009	2009	F4	Nej	Nej	Ja	Nej	attestation of a kiosk computer
Coprocessor-based hierarchical trust management for software integrity and digital identity protection	Wang, L., & Dasgupta, P.	Journal of Computer Security	2008	F4	Ja	Nej	Ja	Nej	protection agains rootkits and other malware by attesting software integrity
A software authentication system for information integrity	Lein Harn, Hung-Yu Lin, Shoubao Yang	Computers and Security	1992	F3	Nej	Nej	Nej	Nej	Gammal artikel om signerad mjukvara.
Network-based root of trust for installation	J. Schiffman, T. Moyer, T. Jaeger, P. McDaniel	IEEE Security & Privacy	2011	F3	Ja	Nej	Ja	Ja	Säker installation över nätverk.
A Memory and Disk Integrity Protection Scheme	Ma, H., Yao, N., Han, Y., Zhao, J., & Gao, Z.	5th International Conference on Internet Computing for Science and Engineering	2011	F3	Ja	Nej	Nej	Nej	integrity protection av data på hårddisken och minnet med hjälp av hashtrees
A secure bootstrap based on trusted computing	Junkai Gu, Weiyong Ji	International Conference on New Trends in Information and Service Science	2009	F3	Ja	Nej	Ja	Nej	secure bootstrap based on TC for static integrity of OSLoaders
A Secure Software Download Framework Based on Mobile Trusted Computing	Wu Jun-jun, Fang Ming-wei, Yu Peng-fei, Zhang Xin-fang	2nd International Workshop on Computer Science and Engineering	2009	F3	Ja	Nej	Nej	Nej	framework for secure download and instalation of software
An Executable Code Authorization Model for Secure Operating System	Chen Zemao, Wu Xiaoping, Tang Weimin	International Symposium on Electronic Commerce and Security	2008	F3	Nej	Nej	Ja	Nej	verifierar att kod inte har ändrats från kompilering till när den ska köras
Extending source code generators for evidence-based software certification	E. Denney, B. Fischer	International Symposium on Leveraging Applications of Formal Methods, Verification and Validation	2007	F1, F2	Nej	Nej	Ja	Nej	Automatisk kodgenerering tillsammans med automatgenererade bevis för korrektheten.

Challenges of Establishing a Software Product Line for an Aerospace Engine Monitoring System	Habli, I., Kelly, T., & Hopkins, I.	11th International Software Product Line Conference	2007	F1, F2	Nej	–	–	–	erfarenheter från mjukvaru-certifiering av system med höga integritetskrav
Towards verifying global properties of adaptive software based on linear temporal logic	Zhao, Y., Li, J., Sun, D., & Ma, D.	International Conference on Advanced Information Networking and Applications	2011	F1, F2	Nej	Nej	Nej	Ja	a temporal logic for verification of adaptive software
Do you trust your compiler? Applying formal methods to constructing high-assurance compilers	James M. Boyle, R. Daniel Resler, Victor L. Winter	High-Assurance Systems Engineering Workshop	1997	F2	Nej	Nej	Nej	Nej	En metod för att skapa en formellt verifierad kompilator.
A Verified Runtime for a Verified Theorem Prover	Magnus O. Myreen and Jared Davis	Lecture Notes in Computer Science	2011	F2	Nej	Ja	Ja	Ja	a formally verified Lisp runtime for the Milawa theorem provers
Automatic Verification of Object Code Against Source Code	Subramanian, S., & Cook, J. V	Annual Conference on Computer Assurance	1996	F2	Nej	Nej	Ja	Ja	automatic verification of object code against source code for small programs
Preventing secret leakage from fork (): Securing privilege-separated applications	Umesh Shankar and David Wagner	IEEE International Conference on Communications	2006	F2	Nej	Nej	Ja	Ja	kollar vilken data skickas från en trusted process till en untrusted childs process by some static code analyses
A method for certifying code in trust-by-policy-adherence	Guo-Sun Zeng Li Li	Journal of Computers	2011	F1	Nej	Nej	Nej	Nej	Genom att kombinera aspect-oriented programming med en viss sorts logik kan de verifiera att koden uppfyller säkerhetspolicyn
Certification: Reducing the hidden costs of poor quality	Jeffrey Voas	IEEE Software	1999	F1	Nej	–	–	–	Översikt, populärvetenskaplig, över testnings-certifierings och kvalitetspapper för mjukvaruutveckling.
Increasing assurance with literate programming techniques	Andrew P. Moore, Charles N. Payne Jr	Annual Conference on Computer Assurance	1996	F1	Nej	Nej	Ja	Nej	En metod för att skriva granskningsbar källkod, där det är lättare att verifiera att den överensstämmer med specifikationen.
Symbol security condition considered harmful	M. Schaefer	Symposium on Research in Security and Privacy	1989	F1	Nej	Nej	–	–	En analys av TCSEC:s krav på formell verifiering av kod.
Software evaluation in high integrity systems	I. Lloyd	Computers and Security	1990	F1	Nej	Nej	–	–	En diskussion ang. utveckling och evaluering av högassuransmjukvara i UK.
Reliability modeling for safety-critical software	N. Schneidewind	IEEE Transactions on Reliability	1997	F1	Nej	Nej	Ja	Ja	Beskriver utveckling av safety-kritisk kod för rymdfärjan.
One in a baker's dozen: Debugging debugging	J. Voas, K. Miller	International Symposium on High Assurans Systems Engineering	2007	F1	Nej	Nej	Ja	Nej	Statistisk metod för att skatta antalet kvarvarande fel vid debugging.
Precise documentation of critical software	D. Parnas, S. Vilkomir	International Symposium on High Assurans Systems Engineering	2007	F1	Nej	–	Ja	Nej	En metod för att beskriva krav på ett entydigt och testbart sätt. Verkar klokt.
Trusting computers through trusting humans: Software verification in a safety-critical information system	A. Adam, P. Spedding	International Journal of Technology and Human Interaction	2007	F1	Nej	Nej	Ja	Nej	Ett teoretiskt ramverk för att förstå hur man kan få trust i automatgenererad kod.
What use is verified software?	J. Rushby	International Conference on Engineering och Vomplex Computer Systems	2007	F1	Nej	Nej	Nej	Nej	Diskussion om hur formella metoder kan användas för att skapa bättre mjukvara.
Contract-based reasoning for verification and certification of secure information flow policies in industrial workflows	J. Hatcliff	Formal Methods in Software Engineering	2008	F1	Nej	Nej	Ja	Nej	Beskrivning av ett projekt för att skapa mer automatiserat stöd vid korrekthetsbevis.

Patterns for secure boot and secure storage in computer systems	H. Löhr, A. Sadeghi, M. Winandy	International Conference on Availability, Reliability and Security	2010	F1	Nej	–	Nej	Nej	Security patterns för secure boot och secure storage.
Trusted product lines	S. Hutchesson, J. McDermid	Information and Software Technology	2013	F1	Nej	Nej	Nej	Nej	Föreslagen metod för att kunna skriva återanvändbar kod inom high-assurance-området.
Conformance testing of software	Scowen, Roger S	Computers and Graphics	1984	F1	Nej	–	–	–	old paper- discusses how to check that software conforms with its specification
Framework for Third Party Testing of Component Software	Yu-Seung Ma, Seung-Uk Oh, Doo-Hwan Bae, and Yong-Rae Kwon	Asia-Pacific Software Engineering Conference and International Computer Science Conference,	2001	F1	Nej	Nej	Nej	Nej	a framework för testing av component-based software som del av certifieringen
Research on Trusted Programming Technology in Information Security	Cai, J.-P., & Xu, W.-Y.	International Conference on Communication Systems and Network Technologies	2012	F1	Nej	Nej	Nej	Nej	software credibility programming
Software assurance using structured assurance case models	Rhodes, T., Boland, F., Fong, E., & Kass, M.	Journal of Research of the National Institute of Standards and Technology	2010	F1	Nej	Nej	–	–	propose structured assurance model for sorfware assurance during development, not only for security assurance.
Design and assurance strategy for the NRL pump	Kang, M. H., Moore, A. P., & Moskowitz, I. S	High-Assurance Systems Engineering Workshop	1997	F1	Nej	Nej	Nej	Nej	software design and assurance strategy
Trustworthy components - Compositionality and prediction	H. Schmidt	Journal of Systems and Software	2003	F1	Nej	Nej	Nej	Nej	Ett arkitekturdefinitionsspråk, kanske lämpat för säkra system.
A formal framework for trust management of service-oriented systems	Liang Zhengping, Liu Xiaoli, Wu Guoqing, Yang Min, Zhang Fan	IEEE International Conference on Service-Oriented Computing and Applications	2007	F1	Nej	–	Nej	Nej	Det är en formell modell för trust.

Referenser

[1] Trusted platform module. *http://en.wikipedia.org/wiki/Trusted_Platform_Module (tillgängligheten kontrollerad i augusti 2013).* Wikipedia.

[2] Never execute bit. *http://en.wikipedia.org/wiki/NX_bit (tillgängligheten kontrollerad i augusti 2013).* Wikipedia.

[3] Intel trusted execution technology. *http://en.wikipedia.org/wiki/Trusted_Execution_Technology (tillgängligheten kontrollerad i augusti 2013).* Wikipedia.

[4] x86 virtualization. *http://en.wikipedia.org/wiki/X86_virtualization (tillgängligheten kontrollerad i augusti 2013).* Wikipedia.

[5] J. Löfvenberg och K. Lundholm. Pålitliga plattformar. Teknisk rapport FOI-R--3136--SE, 2010.

[6] *Terminologi för informationssäkerhet, Utgåva 3.* Nummer SIS HB 550. SIS förlag, 2007.

[7] DoD CIO memo, encryption. *http://iase.disa.mil/policy-guidance/dod-dar-tpm-decree07-03-07.pdf (tillgängligheten kontrollerad i september 2013).*

[8] C.L. Harrold. An introduction to the smite approach to secure computing. *Computers and Security*, 8(6):495–505, 1989.

[9] T.F. Lunt, D.E. Denning, R.R. Schell, M. Heckman och W.R. Shockley. Element-level classification with a1 assurance. *Computers and Security*, 7(1):73–82, 1988.

[10] R.S. Scowen. Conformance testing of software. *Computers and Graphics*, 8(1):5–12, 1984.

[11] J. Rushby. What use is verified software? I: *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, ss 270–276, 2007.

[12] A.P. Moore och C.N. Payne Jr. Increasing assurance with literate programming techniques. I: *COMPASS - Proceedings of the Annual Conference on Computer Assurance*, ss 187–198, 1996.

[13] E. Denney och B. Fischer. Extending source code generators for evidence-based software certification. I: *Proceedings - ISoLA 2006: 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, ss 138–145, 2007.

[14] D.L. Parnas och S.A. Vilkomir. Precise documentation of critical software. I: *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, ss 237–244, 2007.

[15] J.M. Boyle, R.D. Resler och V.L. Winter. Do you trust your compiler? applying formal methods to constructing high-assurance compilers. I: *Proceedings of the High-Assurance Systems Engineering Workshop*, ss 14–24, 1997.

[16] Y. Zhao, J. Li, D. Sun och D. Ma. Towards verifying global properties of adaptive software based on linear temporal logic. I: *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, ss 240–247, 2011.

[17] S. Subramanian och J.V. Cook. Automatic verification of object code against source code. I: *COMPASS - Proceedings of the Annual Conference on Computer Assurance*, ss 46–55, 1996.

[18] J.-J. Wu, M.-W. Fang, P.-F. Yu och X.-F. Zhang. A secure software download framework based on mobile trusted computing. I: *2nd International Workshop on Computer Science and Engineering*, band 2, ss 171–176, 2009.

[19] J. Schiffman, T. Moyer, T. Jaeger och P. McDaniel. Network-based root of trust for installation. *IEEE Security and Privacy*, 9(1):40–48, 2011.

[20] S. Balfe, E. Gallery, C.J. Mitchell och K.G. Paterson. Challenges for trusted computing. *IEEE Security and Privacy*, 6(6):60–66, 2008.

[21] A.-R. Sadeghi och C. Stüble. Property-based attestation for computing platforms: Caring about properties, not mechanisms. I: *Proceedings New Security Paradigms Workshop*, ss 67–77, 2005.

[22] S. Nelson. *Survey of Software Assurance Techniques for Highly Reliable Systems.* NASA contractor report. National Aeronautics and Space Administration, Ames Research Center, 2003.

[23] K.M. Goertzel, T. Winograd, H.L. Mckinley, L. Oh, M. Colon, T. Mcgibbon, E. Fedchak och R. Vienneau. Software Security Assurance: State of the Art Report (SOAR). Teknisk rapport, Information Assurance Technology Analysis Center (IATAC) Data and Analysis Center for Software (DACS), juli 2007.

[24] B. Parno, J.M. McCune och A. Perrig. Bootstrapping trust in commodity computers. I: *IEEE Symposium on Security and Privacy*, ss 414–429. IEEE Computer Society, 2010.

[25] T. Sommestad och J. Hallberg. An overview of trust attestation methods. ej utgiven, 2011.

[26] NIST. Glossary of computer security terms [teal green book]. Teknisk rapport NCSC-TG-004, 1988.

[27] Common criteria part 1: Introduction and general model (v3.1 revision 4). *http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf (tillgängligheten kontrollerad i maj 2013).*

[28] ISO/IEC. Information technology – security techniques – security assurance framework – part 1: Introduction and concepts. Teknisk rapport ISO/IEC TR 15443-1:2012.

[29] R.J. Anderson. *Security engineering – a guide to building dependable distributed systems (2. ed.).* Wiley, 2008.

[30] ISO/IEC. Information technology – security techniques — security assurance framework – part 1: Introduction and concepts. Teknisk rapport ISO/IEC 17000:2004.

[31] TCG. TCG glossary. *http://www.trustedcomputinggroup.org/developers/-glossary/ (tillgängligheten kontrollerad i maj 2013).*

[32] Trusted computing: concepts. *http://www.cs.bham.ac.uk/ mdr/-teaching/modules/security/lectures/TrustedComputingConcepts.html (tillgängligheten kontrollerad i maj 2013).*

[33] G. Coker, J.D. Guttman, P. Loscocco, J. Sheehy och B.T. Sniffen. Attestation: Evidence and trust. I: *International Conference on Information and Communication Systems (ICICS)*, ss 1–18, 2008.

[34] P. Verissimo, M. Correia, N.F. Neves och P. Sousa. Intrusion-resilient middleware design and validation. *Annals of Emerging Research in Information Assurance, Security and Privacy Services*, 2008. Elsevier.

[35] Oxford dictionaries. *http://oxforddictionaries.com/ (tillgängligheten kontrollerad i augusti 2013).* Oxford Dictionaries.

[36] Trusted system. *http://en.wikipedia.org/wiki/Trusted_system (tillgängligheten kontrollerad i augusti 2013).* Wikipedia.

[37] ISO/IEC. Information technology – security techniques – information security management systems – overview and vocabulary. Teknisk rapport ISO/IEC 27000:2012.

[38] FM. Handbok för försvarsmaktens säkerhetstjänst : informationsteknik : H Säk IT. Teknisk rapport, 2006.

[39] Nationalencyclopedin. *http://www.ne.se (tillgängligheten kontrollerad i maj 2013).*

FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1000 anställda varav ungefär 800 är forskare. Detta gör organisationen till Sveriges största forskningsinstitut. FOI ger kunderna tillgång till ledande expertis inom ett stort antal tillämpningsområden såsom säkerhetspolitiska studier och analyser inom försvar och säkerhet, bedömning av olika typer av hot, system för ledning och hantering av kriser, skydd mot och hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets forskningsinstitut
164 90 Stockholm

Tel: 08-55 50 30 00
Fax: 08-55 50 31 00

www.foi.se