Så säkrar Arms trygghetszoner din IoT

IoT har potential att skapa ett enormt värde från alla de realtidsdata som samlas upp från sensorer och inbyggda system i olika typer av nätverk. Men IoT innebär samtidigt en risk. Att praktiskt taget allt går att koppla upp och kontakta betyder att det alltid finns frestande attackmål tillgängliga för nätmedborgare med ont i sinnet.

De kan tillbringa dag och natt med att söka efter sårbarheter. När de väl lyckas med ett intrång kan de därefter sänka ett nät, stjäla data eller utpressa de som äger data.

Attacker genomfördes på mer än 40 procent av alla industriella kontrollsystem bara under första halvåret 2018, enligt en rapport från Kaspersky Labs baserad på en global enkät om hackare och deras aktiviteter.

Visserligen är det primära målet typiskt servrar, men det pågår attacker också på simplare till synes poänglösa system.

VEM SKULLE TILL EXEMPEL kunna tro att en digital termometer i ett akvarium skulle utgöra en potentiell säkerhetsrisk? Men så visade sig faktiskt vara fallet på ett kasino. Angripare utnyttjade det faktum att termometrarna hade ett svagt försvar mot cyberattacker för att släppa in sig själva i nätverket. Med foten innanför dörren kunde de sedan skaffa sig betydligt bättre access, och stötte snart på en viktig kunddatabas, som de kopierade.

Intrånget upptäcktes först senare av en säkerhetskonsult som analyserade nätverksloggar. Hen upptäckte att data skickades till en avlägsen server i Finland via protokoll som brukar används för strömmande media.

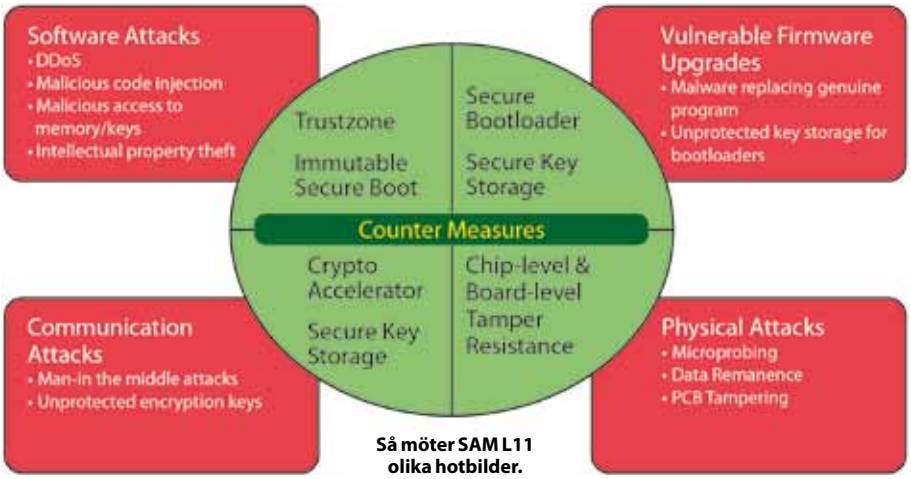
Nätverk på banker har penetrerats via näten som CCTV-kamerorna använder – ironiskt nog installerade för att höja säkerheten. Andra angripare har spionerat på vanliga människors hem via kameror i robotdammsugare.

BÅDA DESSA ATTACKER kunde ha undvikits. De är fullt möjliga att avvärja. Dessa enheter och nätverk skulle kunna skyddas. Här är knepet: att sätta upp ett antal säkerhetslager – en serie barriärer som är svåra och tidskrävande för inkräktare att ta sig genom.

Det angripare gör är att de utnyttjar vissa dåliga designbeslut som mjukvaruutvecklare tyvärr tar alldeles för ofta. Ett vanligt misstag är att använda standardlösenord, exempelvis lösenordet som ligger förprogrammerat i appen som installerar IoT-produkten.

Av Ramanuja Konreddy, Microchip Technology

Ramanuja Konreddy ansvarar på Microchip för just det du ser här – marknadsföring av Microchips strömsnåla hårdvarusäkrade 32 bitars-MCU:er. Hans titel är Senior Product Marketing Engineer.
Sin utbildning till master i elektroteknik fick han på universitetet i San Jose, dit Silicon Valleys många elektronikföretag ofta vänder sig för att rekrytera ny personal.



Så möter SAM L11 olika hotbilder.

En god rekommendation är istället att ordna så att alla IoT-enheter får unika lösenord. Den dag som denna fundamentala strategi är etablerad kommer läget att förbättras, men tyvärr kommer det inte att lösa IoT-säkerhetsproblemet helt och hållet. Angriparna kommer att anpassa sig och hitta mer sofistikerade angreppssätt, liknande de metoder som idag används mot servrar.

Utvecklare av inbyggda system kan tyvärr inte hoppas slippa attacker bara för att de upplever att deras system har litet värde. Som kasinot upptäckte är det möjligt för en angripare att använda godtycklig IoT-enhet som dörr in i nätet.

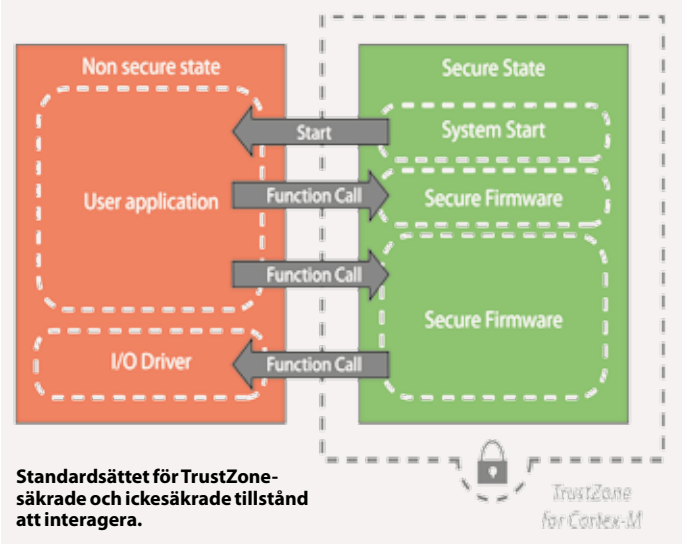
Många olika typer av attacker kan användas mot nätverkade system. Ofta går det att utnyttja brister i programvaran. Om en enhet till exempel får ta emot mer data i ett paket än dess mottagande buffert rymmer kan överskjutande data – noggrant utvald av angriparen – rinna över till näraliggande datastrukturer. Detta kallas för en buffer-overflow-attack.

När någon subrutin senare dyker upp och plockar dessa data från stacken kan det komma i omlopp bland andra rutiner, varvid program kan krascha eller andra oförutsedda saker ske. Processorn kan exempelvis tolka de byte som angriparen planterat som en adress och försöka exekvera kod på den adressen. Med kunskap om enhetens minneslayout kan en angripare på det viset konstruera program som öppnar upp enheten och ger angriparen fria händer. En liknande attack är att skicka inkorrekta data som får de subrutiner som bearbetar dessa data att göra fel på sätt som öppnar sårbarheter.

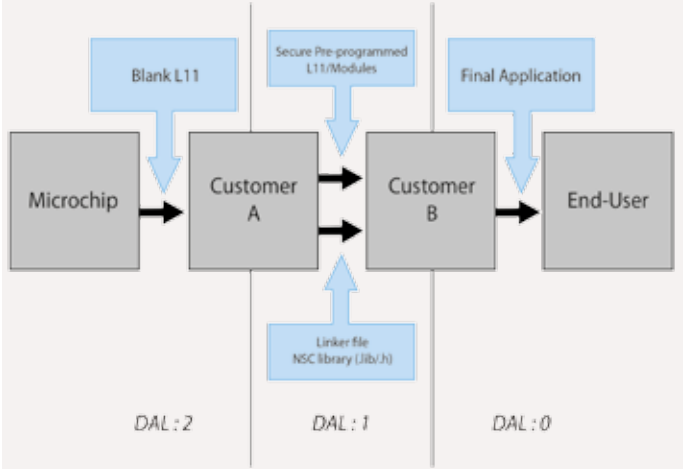
ANDRA ATTACKER FOKUSERAR på kommunikationsprotokoll snarare än systemnära programvara, och försöker överbelasta systemet så att det trampar fel. När enheten därefter försöker återhämta sig kan angriparen finna en öppning där den ge sig själv åtkomst.

Om angriparen har kontroll över näraliggande nätutrustning eller kanske till och med direkt fysisk åtkomst, kan angriparen utge sig för att vara en legitim servrar som enheten vil kommunicera med.

Sådana så kallade man-in-the-middle-attacker kan användas för att analysera de ▶▶



Standardsättet för TrustZone-säkrade och ickesäkrade tillstånd att interagera.



Kund A utvecklar ett säkrat kodbibliotek. Koden läggs i flash och skyddas från åtkomst på Debug Access Level 1. Kund B får endast hantera ickesäkrad kod men kan utnyttja det programgränssnitt som A tog fram, via en länkfil.

data som kommer från enheten för eventuell användning i en framtida attack. Alternativt kan angriparen försöka ladda egen firmware på enheten. När enheten sedan startas om, med ny firmware, kan den fås att göra allt angriparen ber om.

I den bästa av världar skulle systemet avvisa kontaktförsök från maskiner som inte kan dokumentera sina åtkomsträttigheter. Då skulle enheten kunna avvisa all falsk firmware. Den skulle då också omedelbart kunna avslå alla kontaktförsök – och därmed inte binda upp resurser – när den utsätts för en så kallad DOS-attack (denial-of-service). Sådana går ut på att överbelasta systemet genom att översvämma det med datatrafik.

ENHETEN BORDE VÄGRA att skicka känslig information till en maskin som inte autentiserats ordentligt. Längd och dataformat bode undersökas på alla paket som togs emot. Därmed skulle paket i felaktiga format kunna avvisas – paket som skulle kunna bryta sig in i enheten via buffer-overflow-attacker eller kommandoinjektion.

Tyvärr vore det oöverstigligt dyrt att implementera alla dessa preventiva åtgärder i firmware i en stor kodbas, särskilt om den i stor omfattning består av programbibliotek och appar från tredjepart.

Ett mer realistiskt försvar är att dela upp firmware i två separata sektioner. En som kräver hög säkerhet och en som faktiskt kan tillåtas att fallera under ett angrepp eftersom detta inte skulle äventyra systemsäkerheten.

En subrutin som bara slår in temperaturdata i ett JSON-format, för vidarebefordran till en smartphone-app, behöver inte grubbla över informationens säkerhet. Däremot är det bra om en säkrad rutin autentiserar informationen innan den skickas vidare.

Den mjukvara som behöver vara härdad fullt ut för säkerhet, kanske bara utgör en bråkdel av den totala kodbasen. Separeringen är dock endast effektiv om det inte finns några bakdörrar mellan ickesäkrad och säkrad kod. Sådana bakdörrar kan utnyttjas till en så kallad privilege-escalation-attack. Om en sådan attack exempelvis via buffer-overflow lyckas ladda data i ett säkrat minnesområde, och dessa data befordrar angriparen till systemadministratör – så trollas effektivt samtliga skyddsmekanismer bort.

Det är avgörande att säkrade minnesområden isoleras från icke-säkrade. Och en sådan separering kan endast göras helt robust om den implementeras i hårdvara.

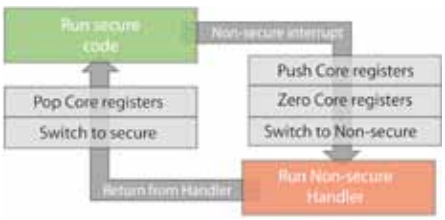
Mikrostyrkretsar – som Microchips familj SAM L11 – använder hårdvaruimplementerad säkerhet baserad på arkitekturen Arm® TrustZone®, förstärkt med ett antal egna skydd mot mjukvarubaserade attacker.

Hårdvaruspärrarna i SAM L11 låter dig konstruera en så kallad root-of-trust som låter dig säkra hela systemet. Detta ger ett heltäckande ramverk för säkerhet.

En root-of-trust använder kryptografiska operationer för att utvidga den säkrade zonen från sig själv till andra delar av systemet. Det gör att kommunikation kan skyddas även om nätverket inte är tillförlitligt.

En integrerad effektiv kryptografisk styrenhet genererar temporära sessionsbegränsade kryptonycklar och utför kryptering och dekryptering.

Styrenheten kan också hjälpa till att verifiera äktheten både hos meddelanden som inkommer från nätet, och äktheten i själva den kod som systemet kör. Det kryptobaserade skyddet kan till och med upptäcka förfalskad hårdvara genom att programvara i den betrodda zonen skickar utmaningar till övriga kort i systemet som endast legitima kort kan formulera korrekta svar på.



Så fungerar avbrottsmekanismen i Cortex M23.

Grunden för root-of-trust i SAM L11 är att startprocessen är säkrad. Den initiala kodsekvens som exekveras vid uppstart är placerad i ROM som inte kan ändras efter tillverkning. Tjänster i ROM-koden verifierar i sin tur ytterligare delar av systemets firmware tills hela uppstartsprocessen är fullbordad.

DET SOM KONKRET VERIFIERAS är att hashsummor för olika segment av firmware är korrekta och att de är konsistenta med en referens-hashsumma kodad i säkringar som brändes vid tillverkningen.

Om en skillnad upptäcks omstartas enheten. Det betyder att om en angripare skulle göra ändringar i den firmware som ligger i flashminnet, skulle enheten vägra starta. Tillverkarens firmware skulle först behöva återställas.

När enheten väl startat och kör firmware som man vet är korrekt, övergår Arm TrustZone på SAM L11 till att upprätthålla en gräns mellan den programvara som är säkrad och den som är en potentiell risk.

TrustZone på Cortex M23-cpu:n i Arm SAM L11 innehåller instruktioner som används för att söka efter problematiska funktionsanrop från ickesäkrad kod till den säkrade domänen.

Arm TrustZone gör det möjligt att med bibehållen systemprestanda dela upp programvaran i domäner på ett sådant sätt att endast säkrad kod kan komma åt vissa minnessegment, viss kringutrustning och viss I/O.

Att Arm TrustZone samlar och skyddar all säkrad kod gör att det blir mycket enklare att utvärdera hur säkert det inbyggda systemet är.

För att särskilja och isolera säker kod från icke-säker kod är SAM L11-minnet uppdelat i regioner som är säkringskonfigurerade att motstå programvaruattacker. Varje försök att komma åt en säkrad minnesregion från en icke-säkrad genererar ett hårdvaruundantag (hard-fault exception) som stoppar åtkom-

sten och dessutom kan utlösa en omstart.

Detsamma händer om kod exekveras när systemet befinner sig i ett tillstånd som inte ska tillåta detta.

Skyddet upprätthålls även under avbrottshantering och debugging.

Implementeringen av Arm TrustZone-tekniken använder bland annat dubbla stackpekare som skiljer mellan säkrad och icke-säkrad exekvering och förhindrar en typ av kompromettering av data på stacken som skulle kunna utföras i en avbrottsrutin utformad för ändamålet.

UNDER FELSÖKNING hanteras säkrad och ickesäkrad kod på olika på olika sätt med hjälp av ett primitiv kallat Debug Access Levels. En utvecklare som arbetar i icke-säkrade sektioner kan inte modifiera säkrad kod och saknar direkt access till dess felsökningsinformation. Det ger en ren modell för separation av ansvar så att endast utvecklare med rätt behörighet kan arbeta med skyddad kod.

Den typiska rollfördelningen är att låta en viss utvecklare ansvara för säkrade applikationer och utveckla biblioteksrutiner med header-filer så att icke-säkrad kod kan anropa rutinerna, som exempelvis kan vara funktioner för kryptering av nätverkspaket som ska skickas över internet.

Den säkrade tillämpningen laddas därefter i ett skyddat minnesområde. Utvecklare utan säkerhetsgodkännande kan utveckla nätverkskod, men endast via biblioteket och länkfilerna och bara via dess programgränssnitt (API). Den säkra koden och dess data utgör en svart låda.

Eventuella försök att ändra säkrad kod stoppas redan vid uppstart eftersom ändringen får hashkoden att sluta att matcha. Allt detta är möjligt via de konsistenskontroller som görs av den ej modifierbara säkrade kod som SAM L11 kör vid uppstart. Om till exempel en pekare ändras så den pekar till ett otillåtet segment, så triggas en undantagssignal.

Även kringutrustning kan designeras som säkrad och ickesäkrad i meningen att endast auktoriserad programvara kan använda eller styra den.

Vad gäller kringutrustning som måste tillhandahålla tjänster till både säkrade och osäkrade regioner, skyddas åtkomsten på motsvarande sätt som funktionsanrop säkras. Icke-säkrad kod begär åtkomst via ett API som skapas av den som utvecklar de säkrade koden. Detta säkerställer att all direkt styrning av kringutrustningen hanteras av autentiserad kod, som kan kontrollera försök till skadlig användning. Till exempel kan ickesäkrad kod tillåtas läsa en timer men hindras från att nollställa den.

EXTERNA ENHETER och delsystem kan endast göra förfrågningar till en mikrokontroller som bifogar ett hashvärde som överensstämmer med kombinationen av förfrågan och ett digitalt certifikat som ligger säkert lagrat on-chip. Detta förhindrar försök från angripare att använda komprometterad kringutrustning för att undergräva systemsäkerheten. Samma teknik används av tillverkare för att hindra att deras enhet används med piratkopierade delsystem. Även om angriparen skulle lyckas hitta och ändra de 256 byte RAM som lagrar nycklar, finns mekanismer i SAM L11 som nollställer nycklar och data när den upptäcker den typen av aktivitet.

Säkerhetsmekanismerna på SAM L11 är sammanlänkade och omfattar såväl fysiskt skydd av minne som separering med hjälp av Arm TrustZone. Detta ger OEM:er möjlighet att utveckla inbyggda system som är säkrade i sin helhet. Därmed förhindras att OEM:ernas applikationer blir den svaga länken för säkerheten i ett IoT-system. ■