Institut für Angewandte Forschung Hochschule Offenburg

forschung im fokus

Ausgabe Nr. 20 / 2017

Hannover Messe-Publikumsmagnet & Fußball-Vizeweltmeister *Sweaty* bereitet sich auf die WM in Japan vor

ivESK – INSTITUT FÜR VERLÄSSLICHE EMBEDDED SYSTEMS UND KOMMUNIKATIONSELEKTRONIK

Das „Internet der Dinge“ durchdringt die industriellen und persönlichen Anwendungen zunehmend. Hierzu zählen beispielsweise Smart-Metering und Smart-Grid, Industrie- und Prozessautomation, Car-to-Car-, bzw. Car-to-X-Kommunikation, Heim- und Gebäudeautomation, Telehealth- und Telecare-Anwendungen. Die drahtgebundene und drahtlose Vernetzung von Embedded Systemen und deren Anbindung als sogenannte cyberphysische Systems (CPS) spielen hierbei eine immer wichtigere Rolle. Da auch immer mehr Systeme funktionskritische Aufgaben autonom übernehmen, gewinnen Zuverlässigkeit und Sicherheit immer mehr an Bedeutung. Entsprechend müssen die Aspekte der Datensicherheit und der Privatsphäre (Privacy) ebenfalls berücksichtigt werden.

Diesen Themen widmet sich das Institut für verlässliche Embedded Systems und Kommunikationselektronik (ivESK) an der Hochschule Offenburg, das im Herbst 2015 von Prof. Dr.-Ing. Axel Sikora und Prof. Dr. rer. nat. Dirk Westhoff gegründet wurde, um die bislang sehr erfolgreichen Forschungs- und Entwicklungsarbeiten in den Laboren der beiden Professoren weiterzuentwickeln und gemeinschaftlich neue Möglichkeiten zu erschließen.

Im Februar 2016 wurde das Institut für sein Automated Physical Testbed (APTB) auch als einer der „100 Orte für Industrie 4.0“ in Baden-Württemberg ausgezeichnet, da dort Unternehmen und Forschungseinrichtungen ihre Kommunikationslösungen für die industrielle Kommunikation automatisiert testen können. Das APTB wurde zudem als Testzentrum im Rahmen der „I4.0 Testumgebungen für KMU - I4KMU“ im Rahmen des BMBF-Programms registriert und konnte hier bereits zwei Projekte im Rahmen dieses Programms gewinnen.

Am Institut arbeiten gegenwärtig 12 Vollzeitmitarbeiter sowie etwa ebenso viele Studierende, wobei aufgrund der positiven Projektlage noch einige Projekt- und Promotionsstellen offen sind. Weitere Kandidaten für Tutorentätigkeiten und Abschlussarbeiten sind gern gesehen.

Institutsleitung
Prof. Dr.-Ing. Dipl.-Ing. Dipl. Wirt.-Ing. Axel Sikora

Merkelphone für Industrie 4.0

Prof. Dr.-Ing. Dipl.-Ing. Dipl. Wirt.-Ing. Axel Sikora, Dipl.-Ing. (FH) Rico Werner

Angesichts aktueller Entwicklungen und Trends im Bereich Embedded, Cloud- und Mobile Computing für das Internet of Things (IoT), der allgegenwärtigen Systemvernetzung mit dem Internet und der damit einhergehenden Steigerung der Anforderungen nach immer mehr parallel auszuführenden Aufgaben, bekommt das Thema Systemvirtualisierung eine immer größere Bedeutung. In diesem Artikel wird ein aktuelles Projekt vorgestellt, das eine Realisierung von virtualisierten Systemen unter Nutzung eingebetteter Systeme ermöglicht.

Facing the developments and trends around embedded, cloud and mobile computing for the Internet of Things (IoT) the increasing connectivity of devices with the Internet leads to a rapidly growing demand of execution of parallel tasks. System virtualization is a promising approach to answer this demand. This paper presents a project realizing one of the manifold approaches to include embedded systems into this virtualization.

Virtualisierung

Die Virtualisierung von Rechnerplattformen ist im Bereich der herkömmlichen PCs und Server ein seit Jahren verbreiteter Ansatz, um eine Unabhängigkeit der einzelnen virtuellen Instanzen voneinander zu erreichen, ohne einen signifikant erhöhten Hardwareaufwand zu bezahlen. Diese Unabhängigkeit kann den Einsatz unterschiedlicher Betriebssysteme, die bessere Nutzung der Ressourcen durch die Nutzung der gemeinsamen Hardware oder auch die Schaffung von, in Bezug auf Security und Privacy unabhängigen Instanzen erlauben.

In ersten Ansätzen ist nun auch zu sehen, dass dieser Virtualisierungsansatz auch in nicht so leistungsfähigen Rechnerplattformen, wie z.B. Smartphones, Eingang findet. Der in Deutschland wahrscheinlich bekannteste Ansatz findet sich im sogenannten „Merkel-Phone", bei dem die eine Rechnereinheit in zwei virtuelle Instanzen aufgeteilt wird. Dies erlaubt den kombinierten Betrieb einer abgesicherten Instanz, in der zuverlässige und geschützte Dienste ausgeführt werden können, und einer offenen und flexiblen Instanz, auf der einfach und flexibel Updates, Erweiterungen oder Installationen von weiteren Anwendungen (Apps) vorgenommen werden können, die aber ungeschützt ist. Solche Separations- bzw. Isolationskernel bieten gegenüber monolithischen Betriebssystemen wesentliche Vorteile in Bezug auf die Sicherheit. Monolithische Betriebssysteme, wie z. B. Linux oder Android, beherbergen alle wichtigen Systemkomponenten bzw. Funktionen im Kernelbereich und trennen hiervon nur die Nutzerapplikationen ab. Dies bedeutet, dass bei einem erfolgreichen Angriff auf Kernelebene, z.B. auf einen Treiber im privilegierten Modus, oft das gesamte System übernommen werden kann. Hybrid- oder auch Makrokernel versuchen, dieses Problem zu entschärfen, indem sie Teile aus dem Kernelbereich entfernen, die nicht privilegiert laufen müssen. Ein Mikrokernel hingegen besitzt nur rudimentäre Funktionen im Kernelbereich, die in der Regel die Speicher und Prozessverwaltung betreffen. Alle anderen Komponenten werden separiert bzw. isoliert im Nutzermodus ausgeführt, weswegen man hier auch von Separations- bzw. Isolationskerneln spricht. Mit diesen lassen sich z. B. Treiber, Netzwerk, Benutzerprogramme oder betriebliche Anwendungen einfach voneinander trennen, sodass bei einem erfolgreichen Angriff auf einen Bereich nicht unmittelbar auf andere Bereiche übergegriffen werden kann. Spezielle Bereiche, wie z. B. kryptografische Operationen, lassen sich so stark isolieren, dass Angriffe auf sie „quasi" ausgeschlossen werden können. Hierzu müssen die verschiedenen Instanzen streng voneinander unabhängig sein. Insbesondere darf kein ungeschützter Zugriff auf die abgesicherte Instanz möglich sein. Die Effizienz der Virtualisierungsplattform muss hoch sein, um auch auf Plattformen mit begrenzten Ressourcen ausreichend performant betrieben werden zu können. Eine Virtualisierungslösung für Embedded-Plattformen für Anwendungen in der Automatisierung wird derzeit im Projekt „Plattformvirtualisierung für sichere und erweiterbare Rechnerknoten in der Automatisierungstechnik (VirtuAut)" am ivESK erarbeitet.

Referenzen/References:
[[1] J. Liedtke: "Improving IPC by Kernel Design", ISBN:0-89791-632-8, SOSP ,93, 05.-08.12.1993
[2] The Linux Foundation: "Xen Project", https://www.xenproject.org/
[3] Siemens: "Jailhouse", https://github.com/siemens/jailhouse
[4] H. Härtig, M. Hohmuth, J. Liedtke, S. Schönberg, J. Wolter: " The Performance of µ-Kernel-Based Systems", ISBN 0-89791-916-5, SOSP'97, Saint-Malo, France, (5.-8.10.1997)

Stand der Technik

Als Softwaregrundlage für virtualisierte Systeme sind bereits mehrere Ansätze und Produkte bzw. Open-Source-Projekte verfügbar, die einer eingehenden Analyse auf ihre jeweiligen Fähigkeiten in Bezug auf die Verwendbarkeit im vorliegenden Projekt VirtuAut unterzogen wurden. Neben Systemen, die auf ein Betriebssystem wie Linux zur Hardwareverwaltung setzen, basiert ein großer Teil der angebotenen Softwarelösungen auf einem Mikrokernelkonzept mit jeweils verschiedenem Leistungsumfang und Lizenzmodellen. Hierzu zählen der L4-Mikrokernel [1] und Xen [2]. Einen anderen Ansatz, der auf die Nutzung spezieller Hardware in modernen Mikroprozessoren setzt, verfolgt das Projekt Jailhouse [2].

Wenngleich abgesicherte Systeme bereits erfolgreich unter L4-Mikrokernen realisiert wurden, bringt dieser Ansatz architekturbedingt auch eine Reihe von Nachteilen mit sich, die sich besonders im Automatisierungsumfeld hinsichtlich der Zykluszeiten negativ auswirken. Dies ist u. a. darin begründet, dass die Treiber für den Gerätezugriff bei diesem Konzept im User Space laufen und dadurch zu höheren Latenzzeiten beitragen. Ein Virtualisierungssystem mit z. B. Linux und dessen monolithischem Kernel als zentralem Hypervisor, in dem die Gerätetreiber direkt im Kernel Space laufen, reduziert die Zykluszeiten zwar, bringt aber hinsichtlich der Sicherheit des Grundsystems und damit aller virtualisierten Gastsysteme zusätzliche Herausforderungen mit sich. Durch Abwägung der zu erwartenden Vor- und Nachteile aller in Frage kommenden Lösungen und Projekte wurden die zwei Kandidaten Xen und Jailhouse zur näheren Untersuchung herangezogen. Nach umfangreichen Messungen und einer Gegenüberstellung beider Systeme fiel die Wahl auf Jailhouse für die weiteren Entwicklungsschritte, da hier die Vorteile für das angestrebte Einsatzfeld überwiegen.

Jailhouse

Das Open-Source-Projekt Jailhouse, das aus dem Automatisierungsumfeld von Siemens entstanden ist, ist eine vielversprechende Lösung. Jailhouse bezeichnet sich selbst als „partitioning Hypervisor based on Linux“ und verfolgt den Ansatz, allen jeweiligen Gastsystemen Teile der Systemhardware statisch, d. h. über deren gesamte Laufzeit, zuzuteilen. Das betrifft die Vergabe sowohl der CPU-Kerne als auch der Speicherbereiche und der Peripheriehardware an die Gastsysteme, die sogenannten „Cells“, wie in Abb. 1 zu sehen. Die Verwaltung der Hardware, deren Zuteilung an die Cells sowie deren Start, Stop und Überwachung der Cells während der Laufzeit übernimmt hierbei der Jailhouse Hypervisor. Dieser läuft unabhängig und ohne eigenes Betriebssystem (Kernel) auf der Hardware, benötigt jedoch für seinen eigenen Start und alle Verwaltungsaufgaben die Hilfe eines privilegierten Gastsystems, der sogenannten „Root Cell“, die durch ein Linux-System bereitgestellt wird.

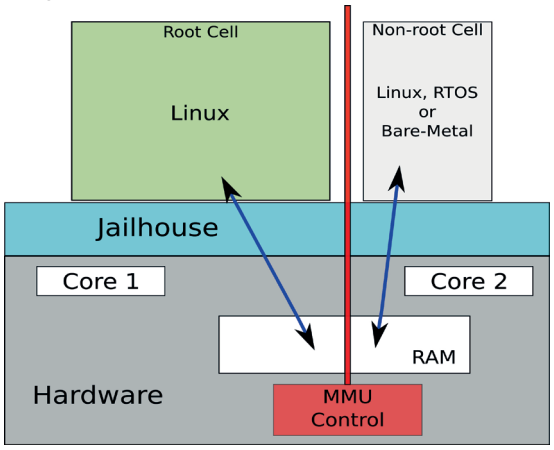

Abb. 1:
Systemübersicht und sichere Trennung der Gastsysteme bei der Verwendung von Jailhouse

Die Vorteile dieses statisch partitionierenden Ansatzes bestehen darin, dass kein zentraler Scheduler existiert, der zusätzlichen Overhead erzeugt, d. h. die Gastsysteme laufen direkt auf der jeweiligen Hardware (CPU), und es werden keine zusätzlichen Latenzzeiten durch einen übergeordneten Scheduler-Prozess eingefügt. Weiterhin bietet Jailhouse gegenüber anderen Lösungen durch seinen Ansatz die Möglichkeit, sogenannte „bare metal guests“, d.h. Gastsysteme ohne eigenes Betriebssystem, wie z. B. FreeRTOS oder eigene nativ in C programmierte Anwendungen, direkt auszuführen. Dies eröffnet insbesondere im Automatisierungsumfeld einen großen Einsatzbereich. Für die Verwaltung der Zugriffsrechte der jeweiligen Gastsysteme auf die Hardwareressourcen wird in Jailhouse von der Möglichkeit Gebrauch gemacht, dies durch integrierte Schutztechnologien im Prozessor bzw. System-on-Chip (SoC) selbst verwalten zu lassen. Moderne multicore ARM-Prozessoren haben bereits diese eingebauten Schutztechnologien, z. B. in Form einer Memory Management Unit (MMU), um Speicherbereiche von Prozessen vor dem Zugriff durch unberechtigte Prozesse zu schützen. Durch die Möglichkeit, Adressbereiche von I/O-Peripherie in den Speicherbereich des Prozessors einzublenden bzw. zu mappen (MMIO –

Memory Mapped I/O) und deren Kombination mit dem in Hardware integrierten Speicherschutz durch die MMU können Zugriffe der Cells auf Peripheriesysteme durch die integrierten Hardwareschutztechnologien überwacht werden. Ein Vorteil, der zugleich Sicherheit und Zeitersparnis gewährleistet. Darüber hinaus bieten die integrierten Schutztechniken auch die Möglichkeit, externe Interrupts direkt an den jeweils zuständigen Prozess oder hier das Gastsystem (Cell) weiterzuleiten, ohne den Umweg über einen zentralen Prozess wie den Hypervisor nehmen zu müssen. Dabei wird vom in den Prozessor integrierten „Virtual Generic Interrupt Controller“ (vGIC) Gebrauch gemacht, der dieses intelligente Routing ermöglicht. Um die genannten Vorteile ausnutzen zu können, ist Jailhouse darauf angewiesen, dass diese Technologien in der Hardware, d. h. im Prozessor bzw. SoC implementiert sind. Das schließt wiederum einige ältere Prozessortypen von der Verwendung für Jailhouse aus. Aktuelle und zukünftige Modelle von Mehrkernprozessoren und SoCs bieten diese Technologie jedoch verstärkt. Sowohl bei Jailhouse wie auch bei den meisten anderen Virtualisierungssystemen müssen die Gastsysteme an den jeweiligen Hypervisor angepasst werden, um dessen bereitgestellte Schnittstellen nutzen zu können. Es ist dadurch nicht möglich, ein unverändertes Betriebssystem unter dem jeweiligen Hypervisor auszuführen.

Inter-Cell-Kommunikation

Der sehr hohe Schutzgrad von Speicher und Peripherie vor dem Zugriff unberechtigter Prozesse durch die integrierte Hardware ist wiederum eine große Herausforderung, wenn Datenaustausch unter den Gastsystemen explizit gewünscht ist. Im Automatisierungsumfeld kann es sich hierbei z.B. um Mess- und Regelungsdaten handeln. Um diesen Datenfluss zu ermöglichen und auch kontrollieren zu können ,wurde in der Grundkonfiguration des Linux-Kernels (Device Tree) der Cell und der Root Cell ein virtuelles Gerät unter Linux erstellt (Abb. 2), das einen gemeinsamen Speicherbereich im RAM verwaltet. Dazu musste auch die Hardware des SoC so konfiguriert werden, dass sie den Zugriff beider Systeme auf diesen Speicherbereich zulässt. Jedes der beiden beteiligten Gastsysteme ist nun über dieses virtuelle Gerät zugriffsberechtigt auf diesen Speicherbereich und es können durch Lese- und Schreibzugriffe Daten ausgetauscht werden. Zur Steuerung des Datenflusses bietet sich die bereits in den Linux-Kernel integrierte Firewall-Funktionalität (netfilter) an. Zu deren Nutzung muss das erstellte virtuelle Gerät über das IP-Protokoll kommunizieren. Dazu ist es notwendig, das gemeinsame virtuelle Gerät als Netzwerkgerät bzw. Netzwerkschnittstelle zu implementieren und dem Kernel bzw. den Anwendungen in den Gastsystemen zu präsentieren. Auf diesem Weg kann nun auf vorhandene Technologien im Linux-Kernel zurückgegriffen werden, die hinsichtlich Security wohl-erprobt sind.

```
/dts-v1/;
/plugin/;
/ {
        compatible = "lemaker,bananapi", "allwinner,sun7i-a20";

        fragment {
                target-path = "/soc@01c00000";
                __overlay__ {
                        #address-cells = <1>;
                        #size-cells = <1>;

                        vpci@2000000 {
                                compatible = "pci-host-ecam-generic";
                                device_type = "pci";
                                bus-range = <0 0>;
                                #address-cells = <3>;
                                #size-cells = <2>;
                                #interrupt-cells = <1>;
                                interrupt-map-mask = <0 0 0 7>;
                                interrupt-map = <0 0 0 1 &gic 0 91 0>,
                                                <0 0 0 2 &gic 0 92 0>,
                                                <0 0 0 3 &gic 0 93 0>,
                                                <0 0 0 4 &gic 0 94 0>;
                                reg = <0x2000000 0x100000>;
                                ranges =
                                        <0x02000000 0x00 0x10000000 0x10000000 0x00 0x30000000>
                                };
                        };
                };
};
```

Abb. 2:
Device Tree Overlay für ein virtuelles Gerät, das zur Kernellaufzeit dynamisch geladen wird

Ergebnis und Ausblick

Im Ergebnis steht eine Virtualisierungslösung zur Verfügung, die sowohl die Anforderungen zeitkritischer Anwendungen hinsichtlich ihrer Verwendung in Automatisierungssystemen erfüllt als auch sehr hohen Schutz vor unberechtigtem Zugriff auf Daten bietet. Die Entwicklung einer überwachten Kommunikationsmöglichkeit von Gastsystemen untereinander innerhalb des virtualisierten Systems eröffnet die Möglichkeit für viele weitere Verwendungen neben der Automatisierungstechnik. Zukünftige Entwicklungen werden noch mehr CPU-Kerne unterstützen. Auch das Debugging solcher virtualisierter Systeme ist ein wichtiges und interessantes Themenfeld, für das bereits der nächste Projektantrag eingereicht wurde.

AUTOREN

Prof. Dr.-Ing. Dipl.-Ing. Dipl. Wirt.-Ing. Axel Sikora
Wissenschaftl. Leiter Institut ivESK
axel.sikora@hs-offenburg.de

Dipl.-Ing. (FH) Rico Werner,
Institut für verlässliche Embedded Systems u. Kommunikationselektronik (ivESK),
rico.werner@hs-offenburg.de

Dank

Dank an das BMWi für die Förderung dieses Projekts im Rahmen des Zentralen Innovationsprogramms Mittelstand (ZIM) unter dem Förderkennzeichen KF-2471323KM4 sowie emtrion GmbH, Karlsruhe und SSV Software Systems GmbH, Hannover für die gute, effiziente und zielgerichtete Zusammenarbeit.