Schlussbericht des Verbundprojekts ENCOPIA für die Siemens AG

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Nummer 16KIS1438 auf die Bekanntmachung 2746 des Bundesministeriums für Bildung und Forschung (BMBF) zur Förderung von deutsch-französischen Projekten zur Cybersicherheit eingereicht.

ENabling COnnected PrIvacy Assurance

Förderkennzeichen: 16KIS1438
Vorhabenbezeichnung: ENCOPIA
Laufzeit des Vorhabens: 01.06.2021 – 31.12.2024

Autor:

Dr. Sven Bauer
Siemens AG
FT RPD CST SES-DE
Otto-Hahn-Ring 6, 81739 München
svenbauer@siemens.com

Kurzdarstellung des Vorhabens

Aufgabenstellung

Das Ziel des Gesamtvorhabens bestand darin, den Weg persönlicher Daten in Geräten des Internet-of-Things (IoT) nachvollziehbar zu machen und so automatisch die Datenschutzeigenschaften solcher Geräte prüfen zu können.

Der Fokus der Siemens-Arbeiten lag insbesondere auf der Erforschung von Möglichkeiten der automatisierten Emulationsumgebungen für IoT-Firmware. Mit derartigen Emulationsumgebungen wird eine genaue Analyse von Datenflüssen in der Firmware von IoT-Geräten erst möglich.

Voraussetzungen, unter denen das Vorhaben durchgeführt wurde

Das Technologiefeld CST (Cybersecurity & Trust) der Siemens AG war bzw. ist an Forschungsprojekten beteiligt, in denen Sicherheitsarchitekturen und Lösungen in unterschiedlichen industriellen Anwendungsgebieten mit Bezug zu eingebetteten Systemen untersucht werden.

Zum Beispiel wurden im BMBF-Projekt SIBASE (2013-2016) Security-Anforderungen, -Bausteine und -Architekturen für sichere eingebettete Systeme der nächsten Generation entwickelt [2]. Im BMBF-Projekt IUNO (2015-2018) wurde das Designprinzip „Security by Design“ für Industrie 4.0 Geräte umgesetzt, von der Security-Analyse über die Definition von Anforderungen bis hin zur Erarbeitung einer Sicherheitsarchitektur und der prototypischen Erprobung in Demonstratoren [3]. Im BMBF-Projekt ALESSIO (2017-2019) wurde ein Konzept für ein updatefähiges Secure Element im industriellen Kontext erarbeitet und auf Basis eines FPGA-basierten System-on-a-Chip (SoC) prototypisch implementiert bzw. in Form eines Demonstrators vorgestellt [4].

Im BMBF-Projekt VE-FIDES (2021-2024) entwickelten Partner aus Industrie und Forschung Lösungen, um die Vertrauenswürdigkeit von elektronischen Komponenten und Geräten durch neue Fingerprinting-Methoden entlang der Lieferkette zu erhöhen. Das BMBF-Projekt APRIORI (2021-2024) zielte darauf ab, einen hardwarebasierten Schutz von privaten Daten in IoT-Geräten zu gewährleisten und die Daten gegen Angriffe abzusichern.

Planung und Ablauf des Vorhabens

Die Arbeit im Projekt ENCOPIA war in acht Arbeitspaketen gegliedert:

- AP0: Projektkoordination
- AP1: Identifizierung geeigneter Methoden für die datenschutzorientierte Binäranalyse
- AP2: Erkennung von datenschutzrelevanten Funktionalitäten und Datenschutzverletzungen
- AP3: Server-seitige Informationsflussanalyse
- AP4: Analyse des Informationsflusses in Firmware für Embedded Devices
- AP5: Rehosting von Firmware mit automatischer Emulator-Erstellung
- AP6: Integration in das Framework zur ganzheitlichen Durchsetzung des Datenschutzes
- AP7: Auswertung und Demonstrator

Siemens beteiligte sich an folgenden (Teil-)Arbeitspaketen:

- AP1: Identifizierung geeigneter Methoden für die datenschutzorientierte Binäranalyse
    - AP1.3: Identifikation und Auswahl von Methoden zur vollständigen System-Informationsflussanalyse
- AP5: Rehosting von Firmware mit automatischer Emulator-Erstellung
    - AP5.1: Design und Anforderungsanalyse für eine System zur Firmwareemulation
    - AP5.2: Modul zur statischen Analyse des Boot-Prozesses zur Ermittlung der Basisparameter
    - AP5.3: Modul zur automatischen Erstellung einer Build-Tool-Chain für die Architektur der zu analysierenden Firmware
    - AP 5.4: Modul zur Evaluierung des benötigten Emulationsmodus
    - AP 5.5: Modul zur Generierung eines “Emulations-Kernels”
    - AP 5.6: Modul zur Generierung eines Dateisystems für die Emulationsumgebung
    - AP 5.7: Kombination aller Module zu einem Framework für automatisierte Emulator-Erstellung

- AP6: Integration in das Framework zur ganzheitlichen Durchsetzung des Datenschutzes
    - AP6.3: Generierung eines Gesamtsystemreports
- AP7: Auswertung und Demonstrator
    - AP7.2: Meilenstein 2
    - AP7.3: Meilenstein 3

Wissenschaftlicher und technischer Stand zu Beginn des Vorhabens

Eine gut funktionierende Hardware-Emulation ist ein äußerst nützliches Werkzeug zum Testen von Firmware für IoT-Geräte. Datenflussanalysen oder abdeckungsgesteuertes Testen sind ohne eine Instrumentierung des zu testenden Codes oder durch Emulation beispielsweise nicht möglich. Eine Instrumentierung ist aufwendig und verändert den zu testenden Code, was die Zuverlässigkeit der Testergebnisse in Frage stellt. Eine Emulation weist diese Nachteile nicht auf.

Insbesondere auch für Sicherheitstests und Datenflussanalysen ist eine Emulation ein wichtiges Hilfsmittel. Sicherheitstests und Datenflussanalysen wiederum dienen dem übergeordneten Projektziel eines verbesserten Schutzes der Privatsphäre von Nutzern von IoT-Geräten.

Eine Emulationsumgebung für IoT-Firmware muss die Eigenschaften der Hardware, auf der die Firmware eigentlich läuft, möglichst exakt nachbilden. Da IoT-Geräte sehr verschiedene Peripherie haben können (Sensoren, Schalter, Netzwerkadapter etc.), kann es keine universelle Emulationsumgebung geben, sondern diese Umgebung muss immer für eine bestimmte Firmware angepasst konstruiert werden.

Ein Ziel des Projektes war daher die Erstellung eines Frameworks, um den Aufbau einer Emulationsumgebung zu erleichtern und so weit wie möglich zu automatisieren. Das heißt, es sollten Software-Werkzeuge entwickelt werden, die durch Analyse einer Firmware die von dieser Firmware erwarteten Peripheriegeräte ermitteln und entsprechende Nachbildungen für die Emulation zur Verfügung stellen können.

Zusammenarbeit mit anderen Stellen

Während der gesamten Laufzeit des Projekts ENCOPIA fanden ein kontinuierlicher Austausch und eine kontinuierliche Zusammenarbeit mit allen Projektpartnern statt. Regelmäßige Seminare wurden organisiert, in denen die Projektteilnehmer aktuelle Ergebnisse vorstellen und sich austauschen konnten.

Es ist beabsichtigt, die Kontakte und Kooperationen, die sich im Rahmen des Projekts ergeben haben, auch zukünftig außerhalb des Projekts weiter fortzuführen und für zukünftige gemeinsame Arbeiten und einen Austausch zwischen akademischer Forschung und industrieller Anwendung zu nutzen.

Eingehende Darstellung

Verwendung der Zuwendung und erzielte Ergebnisse im Einzelnen mit Gegenüberstellung der vorgegebenen Ziele

Das Ziel der Siemens AG im Projekt ENCOPIA bestand in der Implementierung eines Frameworks zur möglichst automatischen Erstellung von Emulationsumgebungen für embedded Firmware.

Das Projekt begann mit der Erhebung von Anforderungen an ein derartiges Framework. Anschließend wurden anhand dieser Anforderungen bereits verfügbare Open-Source-Projekte auf ihre Eignung als Grundlage für eine Emulationsumgebung für IoT-Firmware geprüft.

Im Folgenden wird auf die Arbeiten in den einzelnen Unterarbeitspaketen detaillierter eingegangen.

AP1: Identifizierung geeigneter Methoden für die datenschutzorientierte Binäranalyse

Siemens war nur am Unterarbeitspaket 1.3 mit 1 PM beteiligt.

AP1.3: Identifikation und Auswahl von Methoden zur vollständigen System Informationsflussanalyse

Siemens war an diesem Unterarbeitspaket mit 1 PM beteiligt. Dieser Aufwand fiel für den Austausch mit den Projektpartnern zu Analysemethoden zur Datenflussanalyse an.

AP5: Rehosting von Firmware mit automatischer Emulator-Erstellung

In diesem Arbeitspaket lag mit insgesamt 14 PM der Schwerpunkt der Arbeiten von Siemens.

AP5.1: Design und Anforderungsanalyse für ein System zur Firmwareemulatorerstellung

Im Rahmen dieses Unterarbeitspakets wurde eine Reihe von Anforderungen definiert, die eine für die Zwecke des Projekts brauchbare Emulationslösung erfüllen sollte.

Um auf bestehende Arbeiten wie [1], [2], [3], [4], [5] aufbauen zu können, wurden die Anforderungen als Liste von Kriterien für die Einordnung existierender Emulationswerkzeuge formuliert, anhand derer eine bestehende Arbeit als Grundlage ausgewählt werden konnte.

AP5.2: Modul zur statischen Analyse des Boot-Prozesses zur Ermittlung der Basisparameter

Firmware für embedded Geräte wird typischerweise als einzelne Dateigeliefert, das gepackt wiederum weitere Files enthält. Diese Dateien können wieder ganze Datei-Systeme enthalten. Um an die enthaltenen Daten zu gelangen, müssen diese Dateien als virtuelle Filesysteme von einem Betriebssystem eingebunden und so als zusätzliches virtuelles Laufwerk sichtbar gemacht werden. Diese Schritte sind notwendig, um die Analyse des Boot-Prozesses überhaupt erst zu ermöglichen.

Auch hier wurden existierende Software-Werkzeuge wie [6] und [7] betrachtet und mit Blick auf ihre Eignung für ein Framework zur Emulatorerstellung geprüft.

Der Boot-Prozess eines Linux-Systems ist komplex und wird durch verschiedene Skripte gesteuert. Eine automatische und vollständig korrekte Analyse beliebiger Firmware ist somit eigentlich nicht möglich, allerdings folgt die meiste Firmware bestimmten Konventionen.

AP5.3: Modul zur automatischen Erstellung einer Build-Tool-Chain für die Architektur der zu analysierenden Firmware

Gelegentlich ist es für die Analyse einer Firmware notwendig, weitere Software-Werkzeuge auf dem zu analysierenden System zu installieren. Diese Werkzeuge müssen mit einer für das Zielsystem passenden Tool-Chain erstellt werden.

Es gibt verschiedene Möglichkeiten, das zu erreichen:

1. Aufsetzen einer Cross-Build-Tool-Chain: Bei diesem Ansatz wird beispielsweise auf einem x86-Host Software für eine ARM-Plattform gebaut [8]. Es gibt viele vorgefertigte Cross-Build-Tool-Chains für die gängigsten Host- und Zielsysteme [9]. Schwierigkeiten können auftreten, wenn gegen Libraries des Zielsystems gelinkt werden muss.
2. Nutzen einer (virtuellen) Maschine mit dem Zielsystem: Das Zielsystem wird soweit möglich auf einer (virtuellen) Entwicklungsmaschine installiert.
3. Nutzung von Docker Containern und „binfmt“-Mechanismus, um auf einem Linux-System einen Container für das Zielsystem auszuführen [10]. In diesem Container können dann Entwicklungswerkzeuge für das Zielsystem ausgeführt werden.

Jeder Ansatz hat Vor- und Nachteile. Die Nutzung von Docker-Containern und „binfmt“-Mechanismus ist zwar der fehleranfälligste Ansatz, aber dafür der am leichtesten umzusetzende. Bestandteil dieses Unterarbeitspaketes war, zunächst den Docker-Ansatz zu versuchen, da er meistens schnell zum Ziel führt. Die gelegentlichen Schwierigkeiten sollten mit Verbesserungen in Docker und dem „binfmt“-Mechanismus mit der Zeit verschwinden.

Da ein Sicherheitstester in der Praxis meistens immer wieder ähnliche Zielsysteme untersucht und sich der Docker-Ansatz als schnell und in der Regel zielführend herausgestellt hat, lohnt es sich nicht, den Vorgang der Build-Tool-Chain-Erstellung weitergehend zu automatisieren.

AP5.4: Modul zur Evaluierung des benötigten Emulationsmoduls

Eine verbreitete Software zur Emulation ist QEMU [11]. QEMU unterscheidet zwischen Emulationen im User Mode und im System Mode. Im User Mode können Programme für eine andere CPU-Architektur als die des Host-Systems auf einem Host-System ausgeführt werden. Voraussetzung ist, dass das Gastprogramm das Betriebssystem erwartet, das auf dem Host läuft. Systemaufrufe des Gastprogramms werden von QEMU auf Aufrufe des Host-Systems abgebildet.

Im System Mode wird dagegen eine reine Hardware ohne Betriebssystem emuliert. Der System Mode eignet sich daher als Emulationsumgebungen für vollständige Firmware, weil die ein Betriebssystem enthält.

Da embedded Systeme in der Regel spezialisierte Geräte mit besonderer Peripherie sind, erwartet die Firmware schon beim Starten genau diese Peripheriegeräte, um sie zu initialisieren. Ein Beispiel für Peripherie, die auf einem gängigen Büro-PC nicht anzutreffen ist, sind Sensoren, die feststellen, ob das Gehäuse des Geräts geöffnet ist, um in diesem Fall aus Sicherheitsgründen Teile des Geräts abschalten zu können. Eine Firmware für ein solches Gerät wird schon beim Start prüfen, ob diese Sensoren ansprechbar sind und falls nicht, das Gerät aus Sicherheitsgründen nicht oder nicht vollständig starten. Falls eine Emulationsumgebung also diese Sensoren nicht simuliert, wird die Firmware in dieser Umgebung nicht richtig starten.

Für das Erstellen einer Emulationsumgebung ergeben sich daher zwei Probleme:

1. Das Erkennen von fehlenden Peripheriegeräten, die die Firmware für ein korrektes Funktionieren erwartet.
2. Das Bereitstellen dieser Peripheriegeräte in der Emulationsumgebung. Das kann durch Simulation des erwarteten Verhaltens oder durch Einbindung von echter Hardware in die Emulationsumgebung („hardware in the loop“) geschehen.

Das erste Probleme lässt sich durch das Untersuchen von Log-Files, die beim Starten des Betriebssystem-Kernels angelegt werden, teilweise lösen. Eine genauere Diagnose von Problemen beim Starten der Firmware wird durch die Systemintrospektion der Emulationssoftware PANDA [5] ermöglicht. PANDA baut auf QEMU auf und enthält Erweiterungen, um beispielsweise Systemaufrufe abzufangen. Das macht eine genauere Analyse möglich. Gleichzeitig kann durch das Abfangen von Systemaufrufen und das Einhängen eigener Funktionalität an solchen Stellen das Vorhandensein von Peripheriegeräten simuliert werden. Das heißt, dass man mit diesem Mechanismus auch einen Lösungsansatz für das zweite Problem hat.

Während sich mit PANDA das Problem der Erkennung und Simulation von erforderlichen Peripheriegeräten also „von außen“ lösen lässt, ist ein anderer Ansatz, den beispielsweise das Werkzeug Firmadyne in [2] verfolgt, den Betriebssystem-Kernel der Firmware durch einen speziell instrumentierten Kernel zu ersetzen. Ähnlich wie PANDA fängt der instrumentierte Kernel bestimmte Aufrufe ab, interpretiert sie und leitet sie gegebenenfalls auf eine Implementierung, die das Verhalten einer erwarteten Hardware simuliert.

Es wurde entschieden, dass der PANDA-Ansatz mit einer Instrumentierung „von außen“ gegenüber dem Ansatz einer Instrumentierung des Kernels selbst (also „von innen“) flexibler und einfacher ist.

AP5.5: Modul zur Generierung eines „Emulations-Kernels“

Ziel dieses Unterarbeitspakets war es, die Erstellung von speziell instrumentierten Betriebsystem-Kerneln zu automatisieren. Wie in der Beschreibung des vorherigen Unterarbeitspaktes erläutert, entspricht das dem Ansatz von Firmadyne [2]. Da, wie gerade erklärt wurde, entschieden wurde, dem Ansatz von PANDA [5] zu folgen und eine Instrumentierung des Kernels „von außen“ statt „von innen“ vorzunehmen, hat sich dieses Unterarbeitspaket im Laufe des Projekts als weniger wichtig herausgestellt. Die geplanten Aufwände von 2 PM wurden daher für ein besseres Verständnis der Betriebsystemintrospektions- und Instrumentierungsfähigkeiten von PANDA eingesetzt.

AP5.6: Modul zur Generierung eines Dateisystems für die Emulationsumgebung

Eine gepackte Firmware bringt ihr Dateisystem mit. Es ist allerdings in Linux-System üblich, Peripheriegeräte über das Dateisystem und spezielle Dateien unter dem Verzeichnis /dev anzusprechen. Diese speziellen Dateien, die Geräteanbindungen darstellen, sind nicht notwendigerweise als Dateien im gepackten Dateisystem vorhanden, sondern werden erst beim Starten der Firmware von den entsprechenden Treibern angelegt.

Während „normale“ Dateien also einfach aus der gepackten Firmware übernommen werden können, fällt das Problem der Erstellung der speziellen Dateien unter /dev mit dem in den vorhergehenden Abschnitten beschriebenen Problem der Simulation von Peripheriegeräten zusammen.

AP5.7: Kombination aller Module zu einem Framework für automatisierte Emulatorerstellung

Um die in den vorangegangenen Unterarbeitspaketen erarbeiteten Methoden zu testen, wurde mit PANDA [5] eine Emulationsumgebung für eine Industriesteuerung erstellt.

AP6: Integration in das Framework zur ganzheitlichen Durchsetzung des Datenschutzes

Siemens war an diesem Unterarbeitspaket mit 1 PM beteiligt. Wie in der Teilvorhabensbeschreibung erläutert, ging es um die Unterstützung der anderen Projektpartner, die die wesentlichen Arbeiten in diesem Unterarbeitspaket geleistet haben.

AP7: Auswertung und Demonstrator

Siemens war nur an den Unterarbeitspaketen 7.2 und 7.3 mit jeweils 1 PM beteiligt. Wie in der Teilvorhabensbeschreibung erläutert, ging es vor allem um die Unterstützung der anderen Projektpartner.

Wichtigste Positionen des zahlenmäßigen Nachweises

Der zahlenmäßige Nachweis wird separat übermittelt.

Notwendigkeit und Angemessenheit der geleisteten Arbeiten

Die Arbeiten haben einen wichtigen Beitrag zum verbesserten Untersuchen von Datenflüssen in embedded Firmware, zum verbesserten Testen der Sicherheit von embedded Geräten und damit zu einem besseren Datenschutz geleistet.

Wie bereits erläutert, sind bestimmte Tests und Analysen ohne eine Emulationsumgebung nicht möglich.

Voraussichtlicher Nutzen

Der voraussichtliche Nutzen der Projektergebnisse ist im separaten Erfolgskontrollbericht beschrieben.

Während des Vorhabens bekanntgewordene Fortschritte auf dem Gebiet des Vorhabens bei anderer Stelle

Nachdem in der Zeit vor Beginn des Vorhabens einige wichtige Veröffentlichungen zur Erstellung von Emulationsumgebungen erschienen waren, sind während der Laufzeit des Vorhabens keine grundlegenden Fortschritte bekannt geworden, was unterstreicht, dass das Problem der werkzeuggestützten Erstellung von Emulationsumgebungen für embedded Firmware komplex ist.

Einen guten Überblick über den Stand der Technik zu Beginn des Vorhabens gibt [12].

Erfolgte oder geplante Veröffentlichung der Ergebnisse

Da konkrete Analysen in der Regel an vertraulicher Firmware von Industriesteuerungen durchgeführt wurden, wurde auf Veröffentlichungen verzichtet.

Literaturverzeichnis

[1] D. D. Chen, M. Egele, M. Woo und D. Brumley, „Towards Automated Dynamic Analysis for Linux-Based Embedded Firmware," in *NDSS '16*, San Diego, 2016.

[2] „Firmadyne Surce Code Repository," [Online]. Available: https://github.com/firmadyne/firmadyne. [Zugriff am 3 12 2024].

[3] M. Kim, D. Kim, E. Kim, S. Kim, Y. Jang und Y. Kim, „FirmAE Source Code Repository," [Online]. Available: https://github.com/pr0v3rbs/FirmAE. [Zugriff am 3 12 2024].

[4] M. Messner und P. Eckmann, „EMBA Source Code Repository," [Online]. Available: https://github.com/e-m-b-a/emba. [Zugriff am 3 12 2024].

[5] „Platform for Architecture-Neutral Dynamic Analysis (PANDA)," [Online]. Available: https://panda.re/. [Zugriff am 3 12 2024].

[6] „unblob," [Online]. Available: https://unblob.org/. [Zugriff am 4 12 2024].

[7] „binwalk," [Online]. Available: https://github.com/ReFirmLabs/binwalk. [Zugriff am 4 12 2024].

[8] J. Gosmann, „Jan Gosmanns Blog," [Online]. Available: https://blog.jgosmann.de/posts/2021/02/07/a-guide-to-crosscompiling-applications/. [Zugriff am 4 12 2024].

[9] „Debian Cross Toolchains," [Online]. Available: https://wiki.debian.org/CrossToolchains. [Zugriff am 4 12 2024].

[10] „Docker multi-platform builds,“ [Online]. Available: https://docs.docker.com/build/building/multi-platform/. [Zugriff am 4 12 2024].

[11] „QEMU,“ [Online]. Available: https://www.qemu.org/. [Zugriff am 5 12 2024].

[12] A. e. a. Fasano, „SoK: Enabling Security Analyses of Embedded Systems via Rehosting,“ in *16th ACM ASIA Conference on Computer and Communications Security*, 2021.