Security-Analyse des Sercos III Protokolls durch Analyse der Spezifikation und praktische Tests

Ruben Konrad
secuvera GmbH

Sebastian Fritsch
secuvera GmbH

Zusammenfassung

Anhand von Analysen der Sercos III-Spezifikation wurden potentielle Schwachstellen identifiziert. Die potentiellen Schwachstellen, welche die Nicht-Echtzeitprotokolle betreffen, wurden an drei Sercos III Slaves durch praktische Tests verifiziert. Es wurden weitere produktspezifische Schwachstellen durch Analysen und Tests an den einzelnen Slaves gefunden und verifiziert. Insgesamt wurden fünf Schwachstellen identifiziert, zu denen jeweils ein Lösungsansatz beschrieben wurde.

1 Einführung

Das Thema IT-Sicherheit in Industrieanlagen gewinnt zusammen mit der Vernetzung und Digitalisierung immer mehr Bedeutung. Bereits heute ist es möglich eine Anlage von einem beliebigen Standort aus zu steuern und diese direkt oder indirekt über das Internet zu steuern. Es ist mittlerweile allgemein bekannt, dass die IT-Sicherheit der Feldbusse und Industrial Ethernets, deren Entwicklung teilweise Jahrzehnte zurückliegen, nicht den heutigen Erwartungen entspricht. Es stellt sich die Frage nach vorhandenen Schwachstellen und wie diese behoben werden können, sodass mögliche Angriffe verhindert werden können.

Die Analysen und Tests aus diesem Bericht wurden an *Sercos III* durchgeführt. *Sercos III* ist ein Industrial Ethernet Bussystem außerdem ein bekannter Standard, der historisch speziell für Antriebs- und Steuerungssysteme entwickelt wurde und heute die Anforderungen eines Automation Bus erfüllt. Die Grundlagen des Bussystems werden in einer kurzen Zusammenfassung nachfolgend erklärt.

Im Kapitel *Grundlagen und Sicherheitseigenschaften von Sercos III* sind die Eigenschaften des Bussystems zusammengefasst und die vorhandenen IT-Sicherheitseigenschaften von *Sercos III* aufgeführt. Das anschließende Kapitel *Schwachstellen* beschreibt die Analysen und die Tests für die einzelnen Testgeräte. Im Kapitel *Lösungsansätze* werden einige mögliche Lösungen für die gefundenen Schwachstellen beschrieben.

2 Grundlagen und Sicherheitseigenschaften von Sercos III

Sercos III basiert auf einer Master-Slave-Architektur. Der Sercos Bus hat exakt einen Sercos Master und bis zu 511 Sercos Slaves. Die Sercos Komponenten (Master und Slave) werden über eine vollduplex Punkt-zu-Punkt Verbindung miteinander verbunden, sodass der Betrieb in Ring- oder Doppellinientopologie möglich ist. In Ringtopologie (logischer Doppelring) werden die Sercos Telegramme in entgegengesetzter Richtung vom Master gesendet und passieren jeden Slave einmal bis diese wieder beim Master ankommen. In dieser Topologie kann genau eine Störung zwischen zwei Komponenten kompensiert werden. Im Störungfall (z. B. defektes Netzwerkkabel) erfolgt ein Wechsel in Doppellinientopologie. Nach der Störbehebung kann noch im Betrieb zur Ringtopologie zurückgewechselt werden. In Doppellinientopologie (logischer Ring) sendet der letzte Slave das Telegramm zurück an den Master. Die Topologien können ebenfalls kaskadiert werden. In diesem Fall ist ein Slave wiederum der Master einer weiteren Ring- bzw. Doppellinientopologie.

Auf der Kommunikationsebene erfolgt die Aufteilung zwischen Echtzeitkommunikationskanal und *Unified Communication Channel (UCC)*. Über den Echtzeitkanal erfolgt die Parametrierung (*Service Channel (SVC)*) und Übertragung von Echtzeitdaten (*Real-Time Channel (RTC)*). Der Zugriff auf diesen Kanal ist den Sercos Komponenten innerhalb des Sercos Busses vorbehalten. Ein Zugriff von außen ist im Normalfall nicht möglich. Der *UCC* ist in der Lage alle Protokolle ab ISO/OSI Layer 2 über den Sercos Bus übertragen. Dadurch ist es möglich viele Netzwerk-

protokolle in Kombination mit Sercos III zu verwenden. Eine Ausnahme sind Ethernetframes vom Typ `0x88CD`, welcher für die Übertragung von Sercos Telegrammen reserviert ist. Die Verwendung von *UCC* bietet sich bei Einsatz von weiteren Ethernet- oder IP-basierten Diensten auf den Slaves an, ist allerdings ein optionaler zusätzlicher Kommunikationskanal. Beim Einschalten ist der Sercos Bus im Nicht-Echtzeitbetrieb, d.h. der Sercos Bus verhält sich ähnlich wie ein normales Netzwerk. In diesem Zustand können Netzwerkkomponenten (z. B. PCs) mit den Sercos Komponenten ganz normal über den *UCC* kommunizieren, da dieser dann immer verfügbar ist. Der *UCC* kann für den Echtzeitbetrieb wieder deaktiviert werden, falls er nicht benötigt wird.

Der Bus wird über die *Communication Phases* NRT, CP0, CP1, CP2, CP3 und CP4 hochgefahren. Die initiale Phase beim Einschalten des Systems ist NRT in der nur der *UCC* vorhanden ist. In CP0 erfolgt die Erkennung der Slaves. Die Konfiguration der Echtzeitkommunikationskanäle erfolgt in CP1 und CP2. Außerdem können ab CP2 die Slaves parametriert werden. In CP3 wird die Parametrierung finalisiert und der Sercos Bus ist für die Übertragung von zyklischen Echtzeitdaten bereit. Wenn die gesamte Initialisierung abgeschlossen ist, wird CP4 eingeleitet. In CP4 ist der Sercos Bus voll einsatzfähig und die Applikationen für den Betrieb können gestartet werden.

Sercos III bietet mit dem *S/IP*-Protokoll eine Schnittstelle zu den Sercos Komponenten (Master und Slaves) über den *UCC*. Das *S/IP*-Protokoll umfasst mehrere *S/IP Services*, die sich grob in folgende Kategorien aufteilen lassen: Verbindungsaufbau, Erkennung der Busstruktur und Netzwerkkonfiguration, Identifizierung von Komponenten, Parameterzugriffe und Wartungsfunktionen (Firmwareupdate, Restart). Es gelten prinzipiell die gleichen Lese- und Schreibrechte für den Zugriff auf die Sercos Parameter über die *S/IP Services* wie für den Zugriff über den Echtzeitkommunikationskanal, allerdings mit der Einschränkung, dass kein Zugriff auf die zyklischen Echtzeitdaten über die *S/IP Services* möglich ist. Dem Hersteller bleiben implementierungsspezifische Änderungen und Einschränkungen dieser Rechte möglich. Eine Besonderheit des *S/IP*-Protokolls ist die Möglichkeit, dass in allen *Communication Phases* mit den Sercos Komponenten kommuniziert werden kann, sodass die Slaves unabhängig von der *Communiation Phase* parametriert werden können. Im Vergleich kann die Parametrierung der Slaves über den Echtzeitkommunikationskanal erst erfolgen, wenn der entsprechende *Service Channel* bereitsteht. Außerdem ermöglicht diese Option, dass Parameter auch von außerhalb des Sercos Busses festgelegt werden können. Die Implementierung des *S/IP*-Protokolls ist optional und ist deshalb nicht bei allen Sercos Komponenten vorzufinden.

Das primäre Ziel, der nachfolgenden Funktionen von Sercos III, ist die Gewährleistung des fehlerfreien Betriebs und Schutz vor unautorisierter Bedienung. Diese Sicherheitseigenschaften können sich allerdings auch auf die IT-Sicherheit positiv auswirken.

Es gibt zwei Schutzfunktionen für Parameter. Der Schreibschutz von Parametern ist in dessen Attribut festgelegt. Die Attribute beinhalten weitere Eigenschaften eines Parameter, wie z. B. Speichergröße, Daten- und Anzeigeformat. Diese Eigenschaften werden bei der Implementierung festgelegt und können anschließend nicht mehr verändert werden. Der Schreibschutz kann für die *Communication Phases* CP2, CP3 und CP4 über das Attributfeld einzeln festgelegt werden. So kann zum Beispiel ein Parameter in CP3 und CP4 geschützt werden, aber in NRT, CP0, CP1 und CP2 beschrieben werden. Zu berücksichtigen ist, dass der Schreibschutz in den *Communication Phases* NRT, CP0 und CP1 im Standard nicht festgelegt ist. Damit bleibt es dem Hersteller überlassen, welche Parameter geschützt werden und welche nicht. Außerdem ist zu berücksichtigen, dass es keinen Schutz der verfügbaren Parameter vor lesenden Zugriffen gibt.

Der Passwortschutz schützt eine Liste von Parametern vor einem schreibenden Zugriff, aber ebenfalls nicht vor einem lesenden Zugriff. Die Liste der schreibgeschützten Parameter ist im Parameter *S-0-0279 IDN list of password protected data* gespeichert. Die Liste selbst ist ebenfalls über die Passwortfunktion schreibgeschützt. Der Parameter *S-0-0267 Password* beinhaltet die verschiedenen Funktionen, die für den Passwortschutz nötig sind. So kann über diesen Parameter der Passwortschutz deaktiviert und wieder aktiviert werden. Des weiteren kann bei deaktiviertem Schreibschutz ein neues Passwort festgelegt werden. Um den Schreibschutz zu deaktivieren, muss das gültige Passwort in den Parameter geschrieben werden. Dadurch wird der Schreibschutz für alle Parameter in der Liste aufgehoben, inklusive der Liste selbst. Um den Passwortschutz zu aktivieren, muss entweder ein neues Passwort gesetzt werden oder ein falsches Passwort geschrieben werden. Der Schreibschutz für die *Communication Phases* bleibt von der Passwortschutzfunktion unberührt. Auch wenn der Passwortschutz keine eindeutige Authentifizierung ermöglicht, bietet sich zumindest die Möglichkeit für den Schutz vor einer unbeabsichtigten Parameteränderung.

Eine weitere Schutzmaßnahme ist, dass *RTC* und *UCC* isoliert sind. Dadurch ist kein direkter Zugriff auf den *RTC* von außen möglich. Zusätzlich führt eine Manipulation im *UCC* nicht zwangsläufig zu Problemen mit der Echtzeitkommunikation im *RTC*. Allerdings zeigen die nachfolgenden Tests, dass diese Schutzmaßnahme durch die Verwendung des *S/IP*-Protokolls verringert wird.

3 Schwachstellen

Der Ansatz für die Analysen in diesem Kapitel beruht auf der Annahme, dass in der Spezifikation keine Sicherheitsmechanismen explizit festgelegt, funktionale Teile potentiell fehlerhaft in der Spezifikation definiert oder in *Sercos III*-Produkten implementiert wurden. Die Schlussfolgerung ist, dass es *Sercos III*-Produkte gibt, welche entsprechende Schwachstellen aufweisen.

Ein Teil der Ergebnisse beinhaltet allgemeine und produktunspezifische Schwachstellen auf Basis von Analysen der Spezifikation. Diese Schwachstellen befinden sich damit in allen *Sercos III*-Produkten. Da die Ergebnisse auf theoretischen Überlegungen basieren, wurden diese nach Möglichkeit durch praktische Tests untermauert. Aufgrund der hohen technischen Voraussetzungen für einige Tests, konnten nicht alle potentiellen Schwachstellen verifiziert werden.

Neben den allgemeinen Schwachstellen wurden auch produkt-spezifische Schwachstellen analysiert und getestet. Die Ergebnisse basieren auf praktischen Tests, die an den Testgeräten durchgeführt wurden und betreffen deshalb ausschließlich das untersuchte Produkt. Die Tests konzentrierten sich auf Serverdienste und wurden entsprechend den Methoden des Pentetrationtestings untersucht.

Eine erste Analyse der Spezifikation zeigte, dass eine Manipulation der Daten im *Realtime Channel (RTC)* mit einem hohen Aufwand verbunden wäre, da der Aufwand für die Entwicklung von Tests entsprechend hoch sein würde. Deshalb wurden keine Test für die potentiellen Schwachstellen im *RTC* durchgeführt. Jedoch zeigte sich ebenfalls, dass Angriffe über den *Unified Communication Channel (UCC)* mit relativ geringem Aufwand möglich sind. Deshalb beziehen sich ein großer Teil der Tests auf die Protokolle und Dienste, die im *UCC* verwendet werden können. Die Tests umfassten das *S/IP*-Protokoll, einen FTP- und Telnet-Dienst.

Als Testgeräte kamen drei Leihgeräte von drei verschiedenen Herstellern zum Einsatz. Es wurden zwei Buskoppler (als Buskoppler A und B bezeichnet) und ein Evaluationsboard für die Entwicklung getestet. Alle drei Testgeräte unterstützen das *S/IP*-Protokoll, das im Verlauf dieses Berichts noch eine größere Bedeutung hat. Im Testaufbau kam der *Sercos III Slave Conformizer* [1] als Sercos Master zum Einsatz. Für die Tests wurde der Sercos Bus in Doppellinientopologie mit dem Sercos Master und Sercos Slave (Testgerät) betrieben. An das offene Ende wurde ein Notebook angeschlossen, welches zur Analyse der Schwachstellen verwendet wurde.

In den folgenden Kapiteln werden die Analysen zur *Verschlüsselung und Authentifizierung* für *RTC*, *UCC* und das *S/IP*-Protokoll beschrieben. Außerdem werden Schwachstellen im *UCC* zu den Themen *Neustart eines Sercos Slaves*, *Lesen und Schreiben über S/IP*, *Firmwareupdate von Sercos Slaves*, FTP und Telnet Serverdienste beschrieben. Mit Ausnahme des Firmwareupdates konnten für alle potentiellen Schwachstellen Tests durchgeführt werden.

3.1 Verschlüsselung und Authentifizierung

Die Analysen zur Verschlüsselung und Authentifizierung umfassten die Kommunikationskanäle *RTC*, *UCC* und das *S/IP*-Protokoll. Die Ergebnisse aus den Analysen der Spezifikation waren eindeutig; es gibt keine Verschlüsselung für die Kommunikationskanäle und Protokolle. Dieses Ergebnis bestätigte sich während anderen Schwachstellentests, bei welchen parallel der Netzwerkverkehr aufgezeichnet und anlaysiert wurde. Ebenfalls fehlen Authentifizierungsmechanismen für *RTC*, *UCC* und auch die *S/IP Services*, welche über das *S/IP*-Protokoll angeboten werden. Im Folgenden wird darauf eingegangen zu welchen Problemen die fehlenden Schutzmaßnahmen führen können.

Die Übertragung der Sercos Telegramme basiert auf dem Summenrahmenverfahren. Bei diesem Verfahren hat jeder Slave reservierte Datenfelder, die bei der Konfiguration festgelegt werden. Die Datenfelder von anderen Slaves und dem Master dürfen gelesen werden, aber unter keinen Umständen beschrieben werden. Technisch kann das Schreiben nicht unterbunden werden, weil alle Sercos Telegramme jeden Slave durchlaufen müssen. Da eine Verschlüsselung und Integritätssicherung im *RTC* fehlen, ist es jedem Slave möglich den gesamten Datenverkehr zu manipulieren. Die Schwachstelle kann ausgenutzt werden, sofern es gelingt einen manipulierten Slave in den Sercos Bus einzufügen.

Die Authentifizierung soll unter zwei Aspekten untersucht werden. Der erste Aspekt ist die fehlende Authentifizierung bei der Erkennung und Identifizierung von Slaves in CP0 und CP1. Beim Aufbau der Kommunikationskanäle hat der Master nicht die Möglichkeit die Authentizität eines Slaves zu verifizieren. Dadurch könnte ein Busteilnehmer eingeschleust werden, welcher die Telegramme verändert und dadurch die Kommunikation zwischen Master und Slaves bzw. zwischen zwei Slaves manipuliert. Erforderlich ist, dass der eingeschleuste Busteilnehmer die Echtzeitanforderungen von *Sercos III* erfüllt.

Der zweite Aspekt des *RTC* ist die Querkommunikation (*Direct Cross Communication (DCC)*). Um eine schnellere Übertragung von Daten zwischen zwei Slaves

zu ermöglichen bietet *Sercos III* die Querkommunikation, sodass die Daten nicht erst vom Master abgerufen und an den entsprechenden Slave weitergeleitet werden müssen. Der Mechanismus kann auch für die Kommunikation zwischen Master und Slaves verwendet werden. Auch bei dieser Kommunikationsart können die beteiligten Komponenten sich nicht verifizieren und es besteht dieselbe Problematik wie im Abschnitt zuvor.

Die Komplexität der Angriffsszenarien, die sich aus den genannten Schwachstellen ergeben, ist nicht zu unterschätzen. Der Grund dafür sind die harten Echtzeitanforderungen, die bei der Entwicklung berücksichtigt werden müssen. Es wird ein deterministisches und exaktes Zeitverhalten gefordert, welches bei Nichterfüllung zu einem Ausfall des Systems führen würde. Erfüllt also ein entsprechend manipulierter Slave diese Anforderungen nicht, dann kann der Sercos Bus nicht mehr betrieben werden. Diese Anforderungen sind in erster Linie keine Maßnahme für die IT-Sicherheit, aber trotzdem erschweren sie den Angriff erheblich. Es werden entsprechend gute Kenntnisse der Sercos Spezifikation und Ressourcen für die Entwicklung und Testumgebung des manipulierten Slaves benötigt. Der Aufwand vergrößert sich abhängig vom Ziel des Angriffs. Das Mitlesen der Daten ist deutlich einfacher zu implementieren als eine gezielte Manipulation von Daten. Zusätzlich stellt der physikalische Zugang zum Sercos Bus ein zusätzliches Hindernis dar. Ist ein Angriff erfolgreich, indem ein manipulierter Busteilnehmer eingefügt wird, dann können über die benannten Schwachstellen eine Vielzahl von weiterführenden Angriffsszenarien realisiert werden.

Der *UCC* ist ebenfalls nicht verschlüsselt und wird auch nicht authentifiziert, aber das Problem gestaltet sich etwas anders als beim *RTC*. Im Unterschied zum *RTC* werden im *UCC* keine Nutzdaten direkt übertragen, weil immer zwingend ein anderes Protokoll verwendet werden muss. Der *UCC* verhält sich damit wie ein unsicheres Übertragungsmedium. In Folge müssen die Protokolle, die den *UCC* verwenden, die entsprechenden Schutzmaßnahmen bieten. Diese Forderung gilt insbesondere für das *S/IP*-Protokoll, welches ein Teil der Spezifikation von *Sercos III* ist.

Allerdings bietet das *S/IP*-Protokoll ebenfalls keine Verschlüsselung und Authentifizierung der *S/IP Services*. Die Folgen sind weitreichender als beim *RTC*, weil kein physischer Zugriff auf den Sercos Bus nötig ist. Der Zugriff auf einen Slave kann aus einem normalen IT-Netzwerk erfolgen, sofern dieses mit dem Sercos Bus verbunden und der *UCC* aktiviert ist. Die Verknüpfung der des IT-Netzwerks mit dem Sercos Bus wird im Normalfall über die Steuerung, wie z.B. SPS oder Industrie-PC erfolgen. Da keine Authentifizierung nötig ist kann jeder Netzwerkteilnehmer aus dem IT-Netzwerk auf den Slave über die *S/IP Services* zugreifen. Wie zuvor und in den nachfolgenden Kapiteln beschrieben, können über das *S/IP*-Protokoll Parameter auf den Slaves verändert und Funktionen ausgeführt werden. Damit hat diese Schwachstelle im Vergleich eine deutlich höhere Kritikalität, weil die Ausnutzung über das Netzwerk erfolgen kann.

3.2 Neustarten eines Sercos Slaves

Der Sercos Bus reagiert in den einzelnen *Communication Phases* unterschiedlich auf den Neustart eines Slaves. In den Phasen NRT, CP0 bis CP3 hat ein Neustart des Slaves keine schwerwiegenden Folgen, da in diesen Phasen noch keine Echtzeitdaten übertragen werden und damit der Betrieb noch nicht eingeleitet wurde. Allerdings führt ein Neustart eines Slaves in CP4 in vielen Fällen zu einem gestörten Betrieb. Die möglichen Fälle werden im folgenden Abschnitt beschrieben.

Im Idealfall schließt der Master den Slave von der Kommunikation im Bus aus, indem er die benachbarten Slave anweist, keine Echtzeitkommunikation an den betroffenen Slave zu senden. Das hat zur Folge, dass der Sercos Bus in Doppellinientopologie wechselt, aber weiter betrieben werden kann. Dies gilt natürlich nur für den Fall, dass der Sercos Bus nicht bereits in Doppellinientopologie betrieben wird und der Slave für den Betrieb nicht unbedingt notwendig ist. Im schlechtesten Fall führt der Neustart des Slaves dazu, dass die gesamte Echtzeitkommunikation zusammenbricht und der Sercos Bus in die *Communication Phase* NRT zurückfällt. Ein abrupter Kommunikationsabbruch führt in den meisten Fällen dazu, dass ein Not-Halt durchgeführt wird.

Die Ursachen, die zu dieser Schwachstelle führen könnten, werden in den folgenden Abschnitten anhand der Spezifikation genauer dargestellt.

Ein Slave kann über den *S/IP Reset Service* oder den Parameter *S-1350 Reboot Device* neugestartet werden. Die Spezifikation gibt implizit vor, dass es nicht möglich sein soll, einen Slave im Betrieb neuzustarten. Der Zustand „im Betrieb“ ergibt sich durch den Zustand *Operating level*, welcher im Zustandsautomaten *Sub-Device* definiert ist (Abbildung 1). Der Zustandsautomat *Sub-Device* ist zuständig für die Umschaltung zwischen der Parametrierung (*Parametrization level*) und dem Betrieb einer Applikation mit den dafür benötigten Ressourcen (*Operating level*). Die Umschaltung aktiviert einen Schreibschutz für einige Parameter, sodass diese im Betrieb nicht verändert werden können. Der Zustandsautomat *Sub-Device* ist ein Teil des

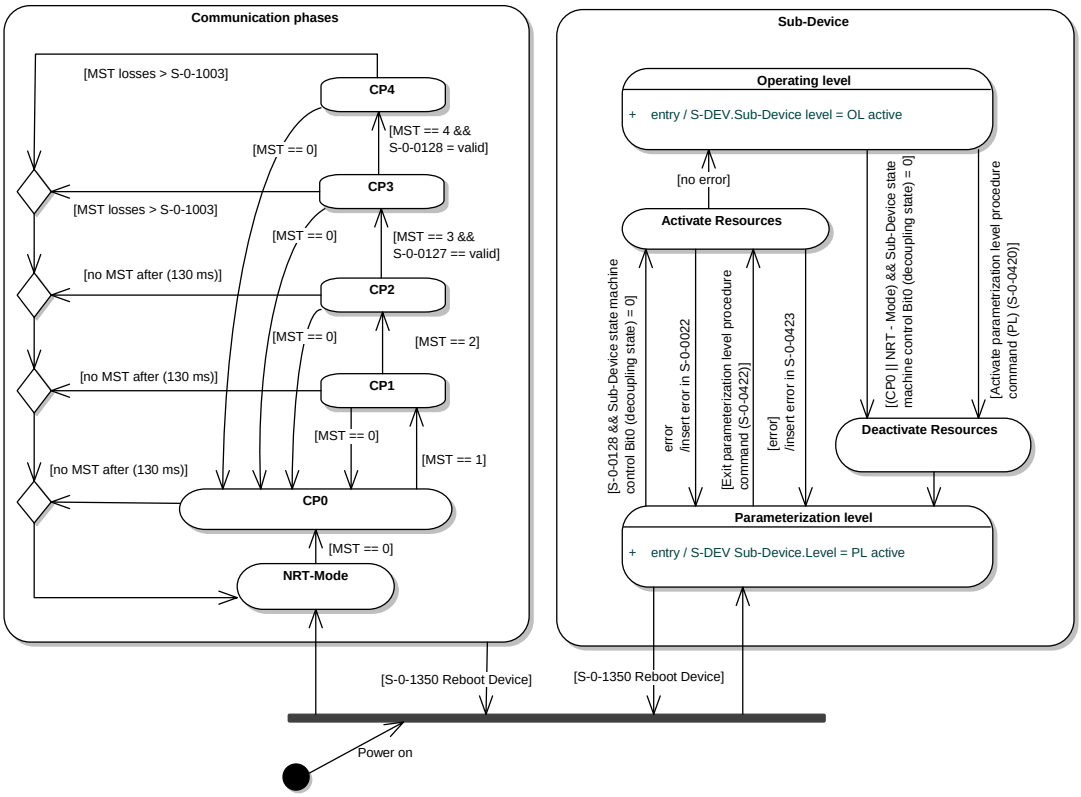

Abbildung 1: Zustandsautomaten: Communication Phase und Sub-Device [3]

Zustandsautomaten *Sercos State Machine*. Der andere Teil des *Sercos State Machine* ist der Zustandsautomat *Communication Phase*. Dieser Zustandsautomat bildet den aktuellen Zustand des Sercos Busses ab.

Die beiden Zustandsautomaten (*Sub-Device* und *Communication Phases*) können entweder gekoppelt oder entkoppelt sein, wobei nicht alle Sercos Komponenten diese Funktion unterstützen. Abbildung 1 zeigt die beiden parallelen Zustandsautomaten. Die Kopplung bzw. Entkopplung erfolgt über den Parameter *S-0-0425 Sub-device state machine control*. Die Kopplung bedeutet, dass zum einen die beiden Substates *CP4* und *Operating level* und zum anderen die Substates *NRT-Mode* und *Parametrization level* miteinander verknüpft sind. Das hat zur Folge, dass der Zustandswechsel von *Parameterization level* zu *Operating level* gleichzeitig durchgeführt wird, wenn der Substate *CP4* erreicht wird. Umgekehrt erfolgt ein Zustandswechsel von *Operating level* zu *Parameterization level*, sobald der *NRT-Mode* aktiv wird. Alle Transitionen können nur erfolgreich durchgeführt werden, wenn die entsprechenden Vorbedingungen erfüllt sind.

Im Gegensatz zu den gekoppelten Zustandsautomaten hängen die inneren Zustände der beiden Zustandsautomaten im entkoppelten Zustand nicht voneinander ab. Die Entkopplung bietet einen Vorteil in Bezug auf die Konfiguration während des Betriebs. Im Betrieb sind viele Parameter schreibgeschützt und können deshalb nicht verändert werden, wie z. B. die Buszykluszeiten, die unter keinen Umständen geändert werden dürfen. Um trotzdem bestimmte Parameter in Spezialfällen verändern zu können, bietet die Entkopplung der Zustandsautomaten die Möglichkeit, dass der Schreibschutz der Applikationsparameter dynamisch zur Laufzeit aufgehoben werden kann. Dazu muss die Applikation auf dem Slave beendet werden und der Zustandsautomat in den Zustand *Parameterization level* überführt werden. Zusätzlich zur dynamischen Aufhebung des Schreibschutzes, hat der Mechanismus hat den Vorteil, dass nicht der gesamte Sercos Bus angehalten und neukonfiguriert werden muss. So können die anderen Sercos Komponenten weiterhin in Echtzeit miteinander kommunizieren.

Wenn ein Hersteller den Zustandsautomaten als Referenz für die Implementierung verwendet, besteht das Potential, dass dadurch eine Schwachstelle implementiert wird. Der Grund dafür ist, dass der Schreibschutz des Parameters *S-1350 Reboot Device* nach Spezifikation „nur“ im *Operating level* aktiv ist und nicht zusätzlich noch in CP4. In

Abbildung 1 sieht man, dass der Parameter *S-1350 Reboot Device* einmal beim Substate *Parametrization level* und einmal beim Zustandsautomaten *Communication phases* vorhanden ist. Die beiden Transitionen sollen anzeigen, dass beim Übergang der jeweiligen Zustände zur Vereinigung (*join* = schwarzer Balken) ein Neustart ausgeführt wird. Die Transition beim *Parametrization level* stellt die gewünschte Funktionalität dar, dass nur beim Konfigurieren des Slaves ein Neustart durchgeführt werden kann, und ein Neustart im Betrieb *Operating level* nicht möglich ist. Der entscheidende Fehler besteht bei der Transition von *Communication phases* zur Vereinigung. Es gibt keine Bedingung, welche die Ausführung der Transition beschränken würde. Folglich kann die Transition unabhängig von jedem Substate im Zustandsautomaten *Communication phases* erfolgen. Die Ungenauigkeit mag unscheinbar erscheinen, hat aber zur Folge, dass bei einer exakten Implementierung anhand der Zustandsautomaten der betroffene Slave in allen *Communication phases* inklusive CP4 neugestartet werden kann. Für den praktischen Einsatz bedeutet das, dass wenn alle Slaves in Echtzeit miteinander kommunizieren, der betroffene Slave neugestartet werden kann. Das System kann darauf mit den eingangs beschriebenen Möglichkeiten reagieren.

Diese Schwachstelle kann über zwei verschiedene Funktionen des Tools *Sercos III S/IP Client* [2] ausgenutzt werden. Das Tool beinhaltet alle Funktionen für den Zugriff auf *S/IP-Services*. Verwendet wurden die Funktionen *Reset Service*, für den direkten Neustart, und die Schreibfunktion *WriteData Service*, über die der Parameter *S-1350 Reboot Device* beschrieben werden kann. Beide Funktionen sind Bestandteil des *S/IP*-Protokolls. Über eine *Reset Service*-Anfrage können bis zu 85 Slaves gleichzeitig neugestartet werden, sodass die Schwachstelle auf mehreren verwundbaren Slaves gleichzeitig ausgenutzt werden könnte. Benötigt wird dazu der *Node Identifier*, welcher (bis auf wenige Ausnahmen) der MAC Adresse des Slaves entspricht.

Es wurde jedes der verfügbaren Geräte einzeln auf diese potentielle Schwachstelle getestet. Dazu wurde das Testgerät in allen *Communication Phases* betrieben und zuerst mit der Funktion *Reset Service* getestet. Als Applikation lief auf dem Testgerät eine einfache Funktion, die einige Echtzeitdaten über den *RTC* übermittelte. Diese Applikation veränderte die Zustandsautomaten nicht, sodass diese dem Auslieferungszustand entsprachen. Anschließend wurde der Test mit der Funktion *WriteData* für die gleichen *Communication Phases* noch einmal durchgeführt. Dazu wurde mit *WriteData* in den Parameter *S-1350 Reboot Device* der Wert `0x0003` geschrieben.

Die Erwartung war, dass das Testgerät, wie nach Spezifikation definiert, in CP1 und CP2 immer neugestartet werden kann. Für CP3 und CP4 war die Erwartung, dass sich das Testgerät über die beschriebene Methode nicht neustarten lässt.

In den Tests zeigte sich, dass alle drei Slaves entsprechend den Erwartungen in CP1 und CP2 neugestartet werden konnten. Das Evaluationsboard konnte weder in CP3 noch in CP4 neugestartet werden und erfüllte damit die Erwartungen. Allerdings konnten, entgegen den Erwartungen, die beiden Buskoppler auch in CP3 und CP4 neugestartet werden. Der Grund dafür könnte ein Implementierungsfehler in der Neustartfunktion sein oder die Buskoppler wurden über die oben beschriebenen Zustandsautomaten im entkoppelten Zustand betrieben, was wiederum zu dem gleichen Fehler führen würde. Unabhängig von der Ursache, stellt dieser Fehler eine Schwachstelle dar, da die Komponenten dadurch in jeder *Communication Phase* über die *S/IP Services* neugestartet werden können. Welche Ursache die Schwachstelle hat, konnte nicht abschließend festgestellt werden, da es sich bei den Untersuchungen der Komponente um eine Black-Box-Analyse handelte.

Mögliche Lösungsansätze für dieses Problem sind im Kapitel 4 beschrieben. Ein Teil des Problems basiert auch darauf, dass der Zugriff auf Parameter nur teilweise eingeschränkt ist. Dieses Problem wird im nächsten Absatz genauer untersucht.

3.3 Lesen und Schreiben von Parametern über S/IP

Die Parameter sind ein wesentlicher Bestandteil in der Funktionsweise von *Sercos III*, da über diese das gesamte Bussystem konfiguriert und betrieben wird. Es gibt mehrere *S/IP Services* für das Auslesen und Schreiben von Parametern, wie z. B. *ReadEverything Service*, *ReadOnlyData Service* und *WriteData Service*. Darüber hinaus existieren spezifische *S/IP Services*, die dafür konzipiert sind eine Funktion auszuführen, die wiederum auf die Parameter zugreift, wie z. B. *SetIp Service* und *Interfaces Service*. Wie bereits beschrieben, können alle *S/IP Services* ohne Authentifizierung verwendet werden. Als Konsequenz können potentiell alle Parameter gelesen und verändert werden. Es gibt allerdings noch einige Einschränkungen beim Lesen und Schreiben von Parametern.

Die erste Einschränkung ist, dass die Parameter über die *S/IP Services* erreichbar sein müssen. Prinzipiell steht es den Herstellern frei die Parameter von der Nutzung über die *S/IP Services* auszuschließen, sodass einige Parameter über diese weder gelesen noch verändert werden können. Anschließend gibt es keine Einschränkungen für den Lesezugriff auf Parameter. Die weiteren Einschränkungen für den Schreibzugriff sind der Schreibschutz für die *Communication Phases* bzw. *Operating level* und *Parametrization level*.

Die letzte Einschränkung bringt die Liste an Parametern, die durch die Passwortfunktion schreibgeschützt sind und erst nach der Freischaltung verändert werden können.

Ein wichtiger Aspekt, der bei den Analysen und Tests berücksichtigt werden muss, sind die zwei unterschiedlichen Arten von Parametern. Es gibt die S-Parameter, welche durch die Sercos Spezifikation festgelegt werden, und die P-Parameter, die von den Herstellern festgelegt werden. Folglich könnte durch umfangreiche Analysen und Tests der S-Parameter festgestellt werden, welche dieser Parameter eine Schwachstelle darstellen. Da es sehr viele S-Parameter gibt, wurden bei den folgenden Analysen und Tests nur einzelne Parameter ausgesucht, die häufig oder immer verwendet werden.

Der erste Test hatte zum Ziel festzustellen, ob die Anzahl der Parameter für das *S/IP*-Protokoll eingeschränkt wurde. Im zweiten Test sollten undokumentierte Parameter gefunden werden. Der dritte Test war für die Untersuchung des *S/IP SetIP Services* und die zugehörigen Parameter, die für die Netzwerkkonfiguration zuständig sind.

Der erste Test diente der Feststellung, ob die Anzahl der Parameter eingeschränkt wurde. Dazu bedurfte es eines individuellen Werkzeugs um mehrere Parameter auf einmal auslesen zu können. Deshalb wurde auf Basis von Scapy [4] ein eigenes Tool entwickelt, mit dem der Test durchgeführt werden konnte. Das Ziel des Tests war es alle Parameter, die im *RTC* verfügbar waren, in einer Liste zu speichern, um anschließend mit dem Tool zu prüfen, ob die Parameter gelesen werden konnten. Für die Überprüfung wurde ein Teil des *S/IP*-Protokolls und der *ReadOnlyData Service* implementiert. Die Annahme war, dass wenn der Parameter über *ReadOnlyData Service* gelesen werden konnte, der Parameter auch für alle anderen *S/IP-Services* verfügbar sein würde. Wenn der Parameter nicht gelesen werden konnte, antwortet das Testgerät mit einer Fehlermeldung. In stichprobenartigen Tests zeigte sich, dass diese Annahme zutraf. Die Testgeräte wurden in jeder *Communication Phase* betrieben und überprüft.

Die Durchführung erforderte die Liste aller Parameter im *RTC*. Dazu wurde der *Sercos III Slave Conformizer* [1] verwendet, der alle Parameter anzeigt, die im *RTC* zur Verfügung stehen. Anschließend wurde diese Liste in das Tool importiert und wie zuvor beschrieben mit der Funktion *ReadOnlyData Service* überprüft. Jedes Testgerät wurde einzeln in jeder *Communication Phase* untersucht. Dazu wurden mit dem Tool die einzelnen Parameter anhand der Liste überprüft.

Die Erwartung war, dass einige Parameter nicht gelesen werden können und dadurch der Zugriff über das *S/IP*-Protokoll eingeschränkt. Bleiben nahezu alle Parameter verfügbar, würde dies bedeuten, dass viele Informationen über den Aufbau und die Struktur des Sercos Busses abgerufen werden können. Parameter, die Aufschluss über Gerätetyp und Netzwerkstruktur geben, sind wertvolle Informationen bei der Planung von Angriffen und sollten deshalb geschützt werden.

Das Ergebnis zeigte, dass alle Parameter von Buskoppler A gelesen werden konnten. Ähnlich verhielt es sich mit dem Evaluationsboard und Buskoppler B. Bis auf einige wenige Parameter konnten alle Parameter gelesen werden. Die Parameter, welche nicht gelesen werden konnten, wurden nach Angaben der Spezifikation für interne Überprüfungsroutinen verwendet. Alle anderen Parameter konnten gelesen werden. Des Weiteren konnten beim Buskoppler A alle Parameter abgerufen werden.

In dem Test konnten wie erwartet über das *S/IP*-Protokoll keine Parameter der zyklischen Echtzeitdaten gelesen oder verändert werden. Folglich sind diese vor diesem Angriffsvektor geschützt. Allerdings zeigte der Test ebenfalls, dass viele Parameter gelesen werden können. Wie zuvor beschrieben können diese Parameter für einen Angreifer eine wertvolle Informationsquelle für weitere Angriffe sein.

Der zweite Test sollte nicht dokumentierte Parameter aufdecken. Der Ansatz war, dass alle möglichen Parameter in allen *Communication Phases* gefunden werden sollten. Da die Anzahl der möglichen Parameter mit den 2^{32} Möglichkeiten zu groß war, wurde eine sinnvolle Untermenge getestet, sodass ein Test nach einigen Stunden beendet war. Die Auswahl wurde jeweils auf die ersten 4096 möglichen S- und P-Parameter beschränkt. Dazu wurde das existierende Tool angepasst. Anstatt einer vorgegeben Liste, überprüfte es sequentiell die 8192 Parameter. Das Ergebnis wurde anschließend mit der Liste aller verfügbaren Parameter im *RTC* verglichen. Es wurden ebenfalls alle *Communication Phases* überprüft. Die Annahme für diesen Test beruht auf der Vermutung, dass es möglicherweise unbekannte Schnittstellen gibt.

Die Testdurchführung war fast identisch mit dem ersten Test. Im Unterschied zum letzten Test wurden aber deutlich mehr Parameter getestet und es wurden keine Listen als Eingabe verwendet.

Die Ergebnisse zeigten, dass Buskoppler B und das Evaluationsboard einen Parameter im *RTC* nicht anzeigten, aber über den *S/IP Service* zur Verfügung stellten. Der Buskoppler B zeigte einen zusätzlichen Parameter, welcher nach Angaben der Spezifikation die Anzahl der fehlerhaften Ethernetframes zählte. Der Parameter kann beschrieben werden, damit der Zähler zurückgesetzt wird. Allerdings

hatte das Evaluationsboard einen Parameter, über den vermutlich Speicherbereiche ausgelesen werden konnten. Ein manueller Test ergab, dass wenn ein Wert in diesen Parameter geschrieben wurde, dann überschrieb der Slave diesen Wert mit einem anderen unbekannten Wert. Da bei gleicher Eingabe meistens der gleiche Ausgangswert angezeigt wurde, war die Vermutung, dass es sich dabei um eine Diagnoseschnittstelle handelte.

In der Praxis wird teilweise vergessen Diagnoseschnittstellen abzuschalten oder sie bleiben absichtlich aktiv, damit der Hersteller z. B. im Garantiefall, das Gerät untersuchen und den Fehler beheben kann. Dabei wird in Fällen davon ausgegangen, dass fehlende Informationen und die Komplexität der Schnittstelle mit Sicherheit gleichgesetzt werden können. Diese Herausforderungen mag einen gewissen Schutz vor einigen Angreifertypen bieten, allerdings kann bei der Entwicklung nicht vorausgesagt werden, welchem Typ von Angreifer das Gerät im Betrieb ausgesetzt sein wird.

Buskoppler A zeigte keine zusätzlichen Parameter an. Allerdings zeigte sich, dass die Komponente sich in einem fehlerhaften Zustand befand, welcher nicht behoben werden konnte. Selbst eine Neuinstallation der Firmware konnte den Fehler nicht beheben. Da die Komponente zu Beginn der Tests einwandfrei funktionierte, muss davon ausgegangen werden, dass die Komponente im Verlauf der Tests in den fehlerhaften Zustand überging. Die exakte Ursache, welcher Tests zu diesem Zustand führte, konnte nicht ermittelt werden, da nur dieses eine Gerät zur Verfügung stand.

Dieses Ergebnis zeigt ebenfalls, dass Tests, welche das Ziel haben Sicherheitslücken aufzudecken, in manchen Fällen auch funktionale Fehler aufdecken. Selbstverständlich ist eine Analyse auf Sicherheitslücken kein Ersatz für funktionale Tests. Da ein vollständig anderer Ansatz und Zielsetzung für eine Sicherheitsanalyse zugrunde liegt, kann diese jedoch als Ergänzung zu funktionalen Tests gesehen werden.

Der dritte Test umfasste die Netzwerkkonfiguration der Testgeräte. Das Ziel war die Veränderung der Netzwerkkonfiguration über den *SetIp Service* und die entsprechenden Parameter. Um die Netzwerkkonfiguration über die Parameter zu ändern, können die Parameter *S-0-1020 IP address*, *S-0-1021 Subnet Mask*, *S-0-1022 Gateway address* und *S-0-1039 Hostname* verändert werden. Diese Parameter stellen eine Speicherfunktion für die Netzwerkkonfiguration dar. Erst durch die Aktivierung über den Parameter *S-0-1048 Activate network settings* wird diese Konfiguration in die entsprechenden schreibgeschützten Parameter (*S-0-1020.0.1 Current IP address*, ...) geschrieben.

Die Durchführung erfolgte für jedes Testgerät mit dem *Sercos S/IP Client*. Zuerst wurde die Netzwerkkonfiguration über den *SetIp Service* geändert und überprüft. Anschließend wurde die Netzwerkkonfiguration mit dem *WriteData Service* über die beschriebenen Parameter verändert. Verändert wurden Subnetzmaske, Hostname, IP- und Gateway-Adresse.

Die Erwartung war, dass die Netzwerkkonfiguration geändert wird und dadurch das Testgerät nicht mehr über die alten Netzwerkeinstellungen erreichbar ist.

Die Ergebnisse aller Testgeräte zeigten, dass die Netzwerkkonfiguration entsprechend den Änderungen übernommen wurde und die Testgeräte nicht mehr erreichbar waren. Das bedeutet, dass alle IP-basierten Dienste der Slaves unter einer neuen IP-Adresse laufen und sich zusätzlich der Hostname geändert hat.

Die Trennung von *RTC* und *UCC* wirken sich in diesem Fall positiv auf die sicherheitstechnischen Eigenschaften von *Sercos III* aus, weil die Änderung der Netzwerkkonfiguration keine Auswirkungen auf die Echtzeitkommunikation hat.

Die ersten beiden Tests für die Feststellung der verfügbaren Parameter zeigte, dass sehr viele Parameter über das *S/IP*-Protokolle zur Verfügung gestellt wurden. Außerdem konnte durch die fehlende Authentifizierung die Netzwerkkonfiguration der Slaves verändert werden und damit bestehende Netzwerkverbindungen zu diesem Slave unterbrochen werden.

3.4 Firmwareupdate von Sercos Slaves

Die Spezifikation definiert die Verwendung von TFTP für das Firmwareupdate von Slaves. Der *UCC* dient auch hier als Übertragungskanal für das Protokoll. Das Trivial File Transfer Protocol (TFTP) ist ein einfaches Protokoll, dass sich auf die Grundfunktion für die Übertragung von Dateien beschränkt. Dadurch fehlen Schutzmaßnahmen wie Verschlüsselung und Authentifizierung, sodass ein Firmwareupdate von jedem Netzwerkteilnehmer, der Zugriff den *UCC* hat, durchgeführt kann. Das kann dazu führen, dass eine manipulierte Firmware auf den Slave installiert werden kann. Das Problem ist in diesem Fall, dass kein sicherer Transportkanal verfügbar ist.

Es muss bedacht werden, dass in vielen Fällen eine Überprüfung der Korrektheit des Firmwareupdates mittels Prüfsummen stattfindet. Das ist vorrangig zum Schutz gegen Datenfehler bei der Übertragung. Es erschwert eine Manipulation der Firmware, ist aber kein wirksamer Schutz dagegen.

Die Untersuchung des TFTP-Dienstes auf den drei Testgeräten ergaben keine unmittelbare Schwachstelle.

3.5 Standard-Serverdienste

In diesem Kapitel sind zwei Tests beschrieben, welche unabhängig von *Sercos III* sind. Sie zeigen allgemeine Probleme beim Betrieb von zusätzlichen Diensten auf Embedded-Geräten. Die Grundlage für die Tests war eine Analyse der verfügbaren Serverdienste auf den Slaves. Dazu wurde ein Portscan bei allen Testgeräten durchgeführt. Auf dem Buskoppler A liefen noch ein Webserver und ein FTP-Server. Auf dem Buskoppler B lief ein Telnet-Server. Nachfolgend sind die Tests für den Telnet- und FTP-Server beschrieben. Der Webserver reagierte nicht auf Anfragen, weshalb dieser nicht weiter untersucht wurde.

Im ersten Test wurde der FTP-Dienst von Buskoppler A untersucht. Der Zugriff war durch eine Login-Maske geschützt. Für den Login wurde ein Benutzername und Passwort benötigt. Durch Ausprobieren konnte ein Standard-FTP-Benutzername, ein vom Hersteller definierter Benutzer und deren Passwörtern erfolgreich erraten werden. Ein Schwäche von vielen FTP-Diensten ist, dass diese bei der Angabe eines nicht-vorhandenen Benutzers einen Fehler zurückmelden und dadurch sehr einfach überprüft werden kann, welche FTP-Benutzer auf dem System vorhanden sind. Der administrative Benutzer `root` war vorhanden, allerdings konnte das Passwort nicht einfach erraten werden.

Deshalb wurde das Firmwareupdate auf vorhandene Benutzernamen und Passwörter untersucht. Die entsprechenden Logindaten wurden über Reverse Engineering extrahiert. Dazu musste das Firmwareupdate mehrmals entpackt und die einzelnen Teile des Firmwareupdates extrahiert werden. Als Tool wurde *binwalk* [5] verwendet, dass die einzelnen Teile identifizieren und extrahieren konnte. Aus den extrahierten Dateien wurden die lesbaren Zeichen herausgefiltert. Mit einer einfachen Textsuche wurde das Passwort des Benutzers `root` gefunden. Allerdings war der Login für den Benutzer gesperrt, sodass keine weitere Schwachstellen ermittelt werden konnten.

Der zweite Test war der Versuch eine Verbindung zum Telnet-Server auf Buskoppler B herzustellen. Der Dienst hatte keine Loginmaske, sodass direkt eine Eingabeaufforderung zur Verfügung stand. Der Befehl „help“ zeigte eine Liste aller möglichen Befehle an. Die unterschiedlichen Befehle sind nachfolgend in unterschiedliche Kategorien unterteilt. Die erste Kategorie von Befehlen zeigte aktuelle Systeminformationen und Systemparameter an. Über die zweite Kategorie konnte die Netzwerkkonfiguration und die Sercos Adresse geändert werden. Die Sercos Adresse wird für die Adressierung im *RTC* verwendet. Die dritte Kategorie umfasste Befehle für das Speichern und Löschen von Parametern im Flashspeicher und zusätzlich zwei Neustartfunktionen. Die Beschreibung besagte, dass eine Neustartfunktion auf einer tieferen Ebene des Buscontrollers ein Neustart durchführt.

Die tiefergehenden Tests beschränkten sich auf die Neustartfunktionen. Wie erwartet konnten auch über diese Funktionen ein Neustart in allen *Communication Phases* durchgeführt werden, wodurch die zuvor beschriebenen Probleme zum Neustart auch für diese Schnittstelle gelten. Weitere Tests wurden an dieser Stelle nicht durchgeführt, da die meisten Funktionen auf Sercos Parameter zurückzuführen sind, die auch über das S/IP-Protokoll ausgelesen und verändert werden können.

Die Beispiele und Tests aus diesem Kapitel zeigen, dass auch in *Sercos III*-Komponenten zum Teil Serverdienste wie Webserver, FTP und Telnet zum Einsatz kommen und folglich auch mit berücksichtigt werden müssen.

4 Lösungsansätze

Ein vollständiger Lösungsansatz für die Kommunikation in Industrial Ethernet und Feldbusse lässt sich nur aus neuen Sicherheitsstandards für die Industrie ableiten. An dieser Stelle sei der Standard *ISA/IEC 62443 Security for industrial automation and control systems* genannt, der sich in Zukunft als Industriestandard etablieren könnte. Der Normteil *62443-4-2 Technical security requirements for IACS components* ist eine Richtlinie für die Sicherheit in Industriekomponenten und bietet damit bereits heute schon Ansätze.

Die Umsetzung von Schutzmechanismen für die Verschlüsselung und Authentifizierung gestaltet sich in einem Bussystem schwieriger als in anderen IT-Systemen. Eine Problematik ist, dass vorhandene Systeme nur begrenzt verändert werden können, da in Embedded-Geräten in den meisten Fällen die Hardwareleistung optimal ausgenutzt wird und keine zusätzliche Rechenleistung vorhanden ist. Durch neuere und leistungsfähigere Hardware könnte die erforderliche Rechenleistung für die Verschlüsselung und Authentifizierung erbracht werden. Allerdings sind in *Sercos III* die Echtzeitanforderungen des Bussystems das größere Probleme. Die Latenz bei der Datenübertragung würde sich erhöhen, da zusätzliche Berechnungen für Verschlüsselungsalgorithmen und Authentifizierung durchgeführt werden müssten. Das würde sich wiederum negativ auf das Echtzeitverhalten auswirken. Auch hier könnten hardwarebasierte Verschlüsselungsalgorithmen das Problem lösen.

Der *UCC* ist von den Echtzeitanforderungen nur bedingt betroffen. Deshalb wäre es mit der entsprechenden Rechenleistung möglich, diesen Kanal zu verschlüsseln. Da

allerdings der *RTC* und *UCC* das gleiche Netzwerkinterface verwenden, bietet sich eine gemeinsame Lösung für *RTC* und *UCC* an.

Eine Lösung für die Schwachstellen des *S/IP*-Protokolls werden als vordringlich gesehen, da sie eine direkte Kommunikation mit den Busteilnehmern ermöglicht und sich auf die Echtzeitkommunikation auswirken kann. Dem Problem kann auf mehrere Arten begegnet werden. Eine Lösung wäre die Einschränkung der Parameter und des Funktionsumfangs, sodass der Betrieb unter keinen Umständen negativ beeinflusst werden kann. Eine alternativer Lösungsansatz ist die Implementierung von Authentifizierungs- und Verschlüsselungsmechanismen, sodass keine unautorisierten Änderungen vorgenommen werden können. Auch in diesem Fall sollte es möglich sein den Umfang der Funktionen und Parameter nur auf benötigte Funktionen und Parameter einzuschränken. Um eine sicherere Authentifizierung zu gewährleisten, dürfen außerdem keine Masterpasswörter eingesetzt werden und Standardpasswörter müssen geändert werden können.

Der Lösungsansatz für die Schwachstelle, die durch die Möglichkeit eines Neustarts in CP4 besteht, umfasst eine allgemeine Lösung in der Spezifikation und eine produktspezifische Lösung durch die Hersteller. In der Spezifikation sollten die zuständigen Funktionen, Parameter und Zustandsautomaten für den Neustart genauer beschrieben werden, um einer fehlerhaften Implementierung vorzubeugen. Zusätzlich sollten die Hersteller bei der Implementierung entsprechende Tests durchführen, damit ein unautorisierter Neustart im Betrieb nicht ausgeführt werden kann.

Eine Schutzmaßnahme, gegen die Manipulation von Firmwareupdates, kann die Verwendung eines Übertragungsprotokolls mit entsprechenden Mechanismen für die Verschlüsselung und Authentifizierung sein. Eine alternative Lösung könnte die Erweiterung der Spezifikation sein. Durch zusätzliche Integritätsprüfungen könnte das TFTP-Protokoll weiter verwendet und trotzdem eine Manipulation des Firmwareupdate verhindert werden. In Embbeded-Geräten ist dafür eine Integritätsprüfung mittels digitaler Signatur üblich.

Die Sicherheit der Serverdienste Telnet und FTP kann nur durch die Verwendung von sicheren Protokollen gewährleistet werden. Eine Sicherung des *UCC* reicht nicht aus, da dabei der Kommunikationsabschnitt über das IT-Netzwerk nicht gesichert ist. Bezogen auf die beiden Protokolle könnten stattdessen z. B. die verschlüsselten Kommunikationsprotokolle SFTP und SSH verwendet werden.

Literatur

[1] Sercos International e.V, *Sercos III Slave Conformizer* URL: https://www.sercos.de/technologie/implementierung/tools (Juli 2018)

[2] Sercos International e.V, *Sercos III S/IP Client* URL: https://www.sercos.de/downloads/tools/ (Juli 2018)

[3] Sercos International e.V., *Sercos III Specification - Generic Device Profile, Version 1.3.1*, März 2013

[4] Scapy, https://github.com/secdev/scapy (Juli 2018)

[5] binwalk, https://github.com/ReFirmLabs/binwalk (Juli 2018)