Bundesamt
für Sicherheit in der
Informationstechnik

ICS-Security-Kompendium

Testempfehlungen und Anforderungen für Hersteller von Komponenten

Stand: 19.11.2014

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
Tel.: +49 22899 9582-0
E-Mail: icssec@bsi.bund.de
Internet: https://www.bsi.bund.de

Inhaltsverzeichnis

1. Einleitung

Dieser Teil des ICS-Security-Kompendium richtet sich an Hersteller von ICS-Komponenten (Industrial Control Systems) und umfasst sowohl Hard- als auch Software-Komponenten. Das Thema Security muss bei Design und Entwicklung von ICS-Komponenten sowie Anlagen/Maschinen berücksichtigt werden.

In diesem Dokument werden Hilfestellungen für den Test der Komponenten und Maßnahmen vorgestellt, um Schwachstellen zu vermeiden und zu erkennen. Eine ausführliche Beschreibung zu den Grundlagen und Bedrohungen von ICS kann [BSI-ICS] entnommen werden.

Weitere Informationen zu den Themen Gefährdungen, empfohlene Maßnahmen und Durchführung von IT-Security-Tests ergänzen die einzelnen Prozessschritte.

1. Assets identifizieren
   Bei Assets kann es sich um eine zu schützende einzelne Komponente, eine Maschine, einen Anlagenteil oder eine gesamte Anlage handeln. Ebenfalls zu den Assets gehören Daten, die zwischen diesen ausgetauscht werden oder auf diesen abgelegt sind, beispielsweise die erforderliche Software oder Rezepturen. Ein ganzes Prozessleitsystem mit den damit verbundenen Geschäftsgeheimnissen und dem geistigen Eigentum können ebenfalls als Asset betrachtet werden.
   Auswahl und Abgrenzung erfolgen jeweils durch den Hersteller bzw. Integrator.
2. Bedrohungen analysieren
   Eine Übersicht der kritischsten Bedrohungen kann [CS-E-TOP10] und [BSI-ICS] entnommen werden. Auf Grund unterschiedlicher Rahmenbedingungen wird eine eigene Bedrohungsanalyse hierdurch nicht ersetzt. Es soll lediglich der Einstieg vereinfacht werden.
3. Relevante Schutzziele ermitteln
   Schutzziele im Sinne dieses Dokumentes können beispielsweise sein
   - Vertraulichkeit von Daten auf den Geräten (z. B. Rezepturen) und bei der Übertragung
   - Integrität von Daten auf den Geräten und bei der Übertragung
   - Verfügbarkeit der Komponenten und Ausfallsicherheit
   - Authentizität von Daten und Anweisungen
4. Bedrohungen analysieren und bewerten
   Eine Analyse und Bewertung von Bedrohungen innerhalb der Komponente(n) erfolgt durch den Hersteller. Der Integrator berücksichtigt die vom Hersteller vorgegebenen Einsatzbedingungen. Die Überprüfung der Security im späteren Betrieb obliegt dem Betreiber gemäß [BSI-ICS] und nach den entsprechenden Vorgaben durch den Integrator.
5. Schutzmaßnahmen und deren Wirksamkeit bewerten
   Kapitel 2 enthält eine Sammlung von Maßnahmen. Bei der Auswahl der Maßnahmen sind diese auf ihre Eignung und Wirksamkeit für die jeweilige Komponente zu prüfen. Ein Beispiel hierfür ist der Einsatz von Antivirensoftware. Wenn eine Komponente dafür keine ausreichenden Ressourcen zur Verfügung stellt, kann diese Maßnahme nicht sinnvoll umgesetzt werden. Gleiches gilt für den Fall, dass Virensignaturen nicht regelmäßig verteilt werden können. Dies hätte eine sehr schlechte Erkennungsrate zur Folge. Die Maßnahme böte dann keinen zusätzlichen Schutz und könnte somit ebenfalls entfallen.
6. Schutzmaßnahmen auswählen und umsetzen
   Die Auswahl und Umsetzung der Maßnahmen innerhalb der Komponente(n) erfolgt durch den Hersteller auf Basis der Vorgaben des Betreibers und oder des Integrators. Dabei müssen die Schutzziele des Betreibers berücksichtigt werden. Es kann daher sein, dass einige Schutzmaßnahmen nur in Zusammenarbeit mit Integrator und Betreiber ausgewählt und umgesetzt werden können.
   Eine Vorauswahl von wirkungs- und sinnvollen Schutzmaßnahmen sollte der Hersteller jedoch bereits bei der Entwicklung berücksichtigen, um Integratoren und Betreiber später besser unterstützen zu können.

7. Security-Test der Komponenten durchführen
   Um die Wirksamkeit der getroffenen Maßnahmen zu prüfen, werden in Kapitel 3 Fragestellungen zu Security-Tests beschrieben. Diese sollen den Hersteller in die Lage versetzen, diese beginnend mit der Entwicklung bis hin zum Systemtest in die Prozesse einzubinden .
8. Regelmäßige Reevaluierung
   Bei Veränderungen an der Komponente oder definierten Rahmenbedingungen sollte in regelmäßigen Abständen eine Prüfung erfolgen, ob die getroffenen Maßnahmen noch ausreichend sind oder nachgebessert werden sollte.

Bei der Integration der Komponenten ist darauf zu achten, dass die angebotenen Security-Mechanismen aktiviert und genutzt werden. Beim Design neuer und Weiterentwicklung vorhandener Komponenten sollte das Thema Security ebenfalls berücksichtigt werden, um auch auf neue Bedrohungen reagieren zu können. Weiterhin wird die Umsetzung des Defense-in-Depth-Konzeptes empfohlen, welches die schichtweise Nutzung von weitergehenden Schutzmaßnahmen vorgibt.

Das Dokument soll den Einstieg in das Thema Security ermöglichen. Viele der Maßnahmen finden sich auch in anderen Standards (z. B. ISO/IEC 62443, IT-Grundschutz usw.) oder Empfehlungen wieder. Dieses Dokument behandelt eine Auswahl wichtiger Maßnahmen, mit denen begonnen werden sollte. Um das Thema „Security“ im Unternehmen fest zu verankern, wird die Nutzung der Standards empfohlen. Daher erfolgt auch eine Zuordnung der vorgestellten Maßnahmen mit diesen Standards, um einen Umstieg auf ein standardkonformes Vorgehen zu ermöglichen.

1.1. Security & Safety

Zu Beginn soll noch eine Abgrenzung zwischen den Begriffen Safety und Security erfolgen. Der Begriff Safety steht dabei für die funktionale Sicherheit der Maschine oder Anlage. Security beschreibt den Schutz von IT-gestützten Systemen gegen absichtliche herbeigeführte oder ungewollte Fehler. Safety-Systeme müssen ebenfalls gegen solche Angriffe geschützt werden.

Der Begriff „Sicherheit“ oder „sicher“ wird in diesem Dokument daher im Kontext der Security verwendet.

1.2. Glossar

- Angriff
  Ein Angriff ist eine vorsätzliche Form der Gefährdung, nämlich eine unerwünschte oder unberechtigte Handlung mit dem Ziel, sich Vorteile zu verschaffen bzw. einen Dritten zu schädigen. [BSI-Glossar]
- Asset
  Assets sind materielle und immaterielle Werte, die bedroht sein können und schützenswert sind. [VDI2182-1]
- Authentifizierung
  Unter einer Authentifizierung versteht man die Prüfung einer Authentisierung, d. h. die Überprüfung, dass ein Kommunikationspartner tatsächlich derjenige ist, der er vorgibt zu sein. [BSI-Glossar]
- Authentisierung
  Unter einer Authentisierung versteht man die Vorlage eines Nachweises eines Kommunikationspartners, dass er tatsächlich derjenige ist, der er vorgibt zu sein. [BSI-Glossar]
- Autorisierung
  Bei einer Autorisierung wird geprüft, ob eine Person, IT-Komponente oder Anwendung zur Durchführung einer bestimmten Aktion berechtigt ist. [BSI-Glossar]
- Bedrohungen
  Eine Bedrohung ist ganz allgemein ein Umstand oder Ereignis, durch das ein Schaden entstehen kann.

Der Schaden bezieht sich dabei auf einen konkreten Wert wie Vermögen, Wissen, Gegenstände oder Gesundheit. [BSI-Glossar]

- Gefährdungen
  Eine Gefährdung ist eine Bedrohung, die konkret auf ein Objekt über eine Schwachstelle einwirkt. Eine Bedrohung wird somit erst durch eine vorhandene Schwachstelle zur Gefährdung für ein Objekt. [BSI-Glossar]
- IT-Sicherheit
  IT-Sicherheit bezeichnet einen Zustand, in dem die Risiken, die beim Einsatz von Informationstechnik aufgrund von Bedrohungen und Schwachstellen vorhanden sind, durch angemessene Maßnahmen auf ein tragbares Maß reduziert sind. IT-Sicherheit ist also der Zustand, in dem Vertraulichkeit, Integrität und Verfügbarkeit von Informationen und Informationstechnik durch angemessene Maßnahmen geschützt sind. [BSI-Glossar]
- Risikoanalyse
  Mit einer Risikoanalyse wird untersucht, wie wahrscheinlich das Eintreten eines schädigenden Ereignisses ist und welche negativen Folgen der Schaden hätte. [BSI-Glossar]
- Robustheit
  Fähigkeit eines Systems, auf Fehler und unvorhergesehene Zustände reagieren zu können, sodass keine fehlerhaften Aktionen durchgeführt werden. [ACATECH-1]
- Schutzbedarf
  Der Schutzbedarf beschreibt, welcher Schutz für die Geschäftsprozesse, die dabei verarbeiteten Informationen und die eingesetzte Informationstechnik ausreichend und angemessen ist. [BSI-Glossar]
- Schwachstelle
  Eine Schwachstelle ist ein sicherheitsrelevanter Fehler. Ursachen können in der Konzeption, den verwendeten Algorithmen, der Implementation, der Konfiguration, dem Betrieb sowie der Organisation liegen. Eine Schwachstelle kann dazu führen, dass eine Bedrohung wirksam wird und eine Institution oder ein System geschädigt wird. Durch eine Schwachstelle wird ein Objekt (eine Institution oder ein System) anfällig für Bedrohungen. [BSI-Glossar]

2. Schutzmaßnahmen

Die folgenden Schutzmaßnahmen richten sich primär an Hersteller, die durch diese im Rahmen der Entwicklung berücksichtigt werden müssen. Es handelt sich um eine sehr generische Auflistung von Maßnahmen, die sich auf sehr grundsätzliche Eigenschaften der Komponenten beziehen.

Es kann der Fall sein, dass bestimmte Maßnahmen in den Komponenten nicht umgesetzt werden können. Als Beispiel sei hier Antivirensoftware genannt. Für die in Speicherprogrammierbaren Steuerungen (SPS) oder Human-Machine-Interfaces (HMI) eingesetzten Betriebssysteme gibt es oft keine Antivirensoftware. In solchen Fällen entfällt die Umsetzung der Maßnahme. Der Hersteller sollte sich dennoch Gedanken dazu machen, ob hierdurch ein erhöhtes Risiko besteht und dann gegebenenfalls alternative Maßnahmen ergreifen.

Die vorgeschlagenen Maßnahmen stellen grundsätzliche Empfehlungen dar und sollen die Wichtigkeit des in der Einleitung beschriebenen Vorgehens verdeutlichen. Der Fokus liegt dabei auf einer Risikoreduktion und einer Hilfestellung. Spezifische Besonderheiten der Komponenten und daraus resultierende Bedrohungen können nur durch den Hersteller bzw. den Integrator identifiziert werden.

Die Maßnahmen beziehen sich auf die in Komponenten eingesetzte Software und deren Konfiguration. Die Eigenschaften der Hardware zum Beispiel in Bezug auf Ausfallsicherheit müssen ergänzend betrachtet werden, wenngleich auch ein Angriff zur Beeinträchtigung der Verfügbarkeit führen kann.

2.1. Organisation, Entwicklung & Planung von ICS-Komponenten

In einem Unternehmen sollte grundsätzlich ein Informationssicherheitsmanagementsystem (z. B. nach ISO/IEC 27001, IT-Grundschutz, ISO/IEC 62443-2-1 „Requirements for an IACS Security Management System“) vorhanden sein. Dies dient dem allgemeinen Schutz des Unternehmens vor Ausfällen oder Schäden. Dazu zählt im Speziellen der Schutz der Entwicklungsabteilung vor Informationsabfluss (z. B. Abfluss von Konstruktionsdaten) und Manipulationen der Entwicklungsdaten (z. B. Einbau von Schadfunktionen). Auch eine sichere Verteilung der Daten (im Sinne der Security) muss im Fokus stehen.

Zusätzlich werden hier einige Aspekte aufgegriffen, die während des Planungs- und Entwicklungsprozesses berücksichtigt werden sollten. Die Maßnahmen dienen als Einstieg in das Thema und sollten sukzessive um weitergehende Anforderungen aus den zuvor genannten Standards erweitert werden. Der Fokus liegt dabei auf den Softwarebestandteilen, die mittlerweile einen beachtlichen Teil der Entwicklungsarbeit beanspruchen. Dazu gehören beispielsweise Aspekte zum Produktdesign, zur Verwendung von Bibliotheken bei der Softwareentwicklung sowie zur Durchführung von Prüfphasen (sog. Reviews) oder Security-Tests.

M 1 Einführung eines Entwicklungszyklus für sichere Software in Komponenten

Eine grundlegende Verbesserung der Sicherheit eines Produkts kann durch das Etablieren eines security-orientierten Entwicklungszyklus (Secure Software Development Lifecycle) erzielt werden. Dazu sollten einheitliche und verbindliche (dem Stand der Technik entsprechende) Vorgaben zur Implementierung gegeben werden wie beispielsweise zur Verwendung von Bibliotheken bei der Softwareentwicklung sowie zur Durchführung von Prüfphasen (sog. Reviews) und Security-Tests.

Weitergehende Informationen hierzu sind unter anderem in den folgenden Dokumenten zu finden:

- BSI-Leitfaden zur Entwicklung sicherer Webanwendungen [BSI-WEB] (die Anwendbarkeit der Empfehlungen beschränkt sich nicht nur auf Webanwendungen)
- ISO/IEC 62443-4-1 „Anforderungen an die Komponentenentwicklung“ [62443-4-1]

Grundsätzlich sollte eine Freigabe des nächsten Entwicklungsschrittes erst erfolgen, wenn zuvor die festgelegten Sicherheitsanforderungen nachweislich erfüllt worden sind.

Aus den genannten Standards sind folgende Maßnahmen besonders wichtig. Sie zielen auf eine frühzeitige Vermeidung von Fehlerquellen und Erkennung von häufig auftretenden Fehlern im Entwicklungsprozess ab.

M 2 Sensibilisierung der Entwickler und Produktverantwortlichen
Entwickler und Produktverantwortliche sollten angemessen bezüglich Sicherheit sensibilisiert werden. Hierzu eignen sich insbesondere Schulungsmaßnahmen. Dabei sollte auf mögliche Angriffe , deren Techniken und Methoden und die entsprechenden Abwehrmaßnahmen eingegangen werden. Als Einstieg bieten sich hier die „ICS TOP 10 Bedrohungen“ [CS-E-TOP10] und „CWE/SANS TOP 25 Most Dangerous Software Errors“ [CWE-TOP25] [SANS-Top25] an.

M 3 Statische Codeanalyse
Im Rahmen der Entwicklung sollten Werkzeuge zur statischen Codeanalyse eingesetzt werden. Diese unterstützen den Entwickler und helfen Fehler zu vermeiden. Eine Übersicht zu verschiedenen Werkzeugen findet sich in [OWASP-1].

Um die Durchsetzung zu garantieren, sollte der Code erst in die Versionsverwaltung aufgenommen werden, wenn er den Richtlinien der statischen Codeanalyse entspricht (sog. Gated Checkin).

M 4 Robustheit der Implementierung
Die eingesetzten Protokoll-Stacks und Anwendungen sollten mit entsprechender Robustheit implementiert werden. Unter Robustheit wird die korrekte Verarbeitung auch nicht protokoll-konformer Daten verstanden. Eine robuste Anwendung sollte nicht-konforme Daten in einer geordneten Form verarbeiten oder verwerfen. Die Umsetzung wird in der Regel durch einen sog. Communication Robustness Test mittels hardwaregestützter Testwerkzeuge überprüft.

Die Einbindung von Tests zur Robustheit von Implementierungen kann direkt als Teil des Buildprocess stattfinden. Nur wenn dies erfolgreich ist, sollte der Code in die Versionsverwaltung aufgenommen werden (vgl. M 3).

Tritt während des Betriebs einer Komponente ein Fehler auf, muss dieser so behandelt werden, dass ein konsistenter und technisch sicherer Zustand der Komponente gewährleistet ist und auch der Schutz der Daten aufrechterhalten wird.

Eine Komponente ist in einem inkonsistenten Zustand, wenn sie aufgrund einer Schwachstelle in einen unerwarteten Zustand überführt wird und dadurch Daten unkontrolliert verarbeitet werden (z. B. keine Fehlermeldung bei erfolgloser Speicherung von Daten) (vgl. [BSI-GS] M 4.395).

Eine ausführliche Liste an möglichen Testszenarien für Ethernet-basierte Komponenten bietet [ISA-EDSA]. Für PROFINET hat beispielsweise die PROFIBUS Nutzerorganisation e.V. (PNO) ebenfalls Dokumente erstellt [PNO-1].

M 5 Identifikation sensibler Daten und Kommunikationsbeziehungen
Bei der Konzeption von Maschinen/Anlagen sollte für die internen und insbesondere externen Kommunikationsverbindungen eine Schutzbedarfsanalyse durchgeführt werden. Kritische Kommunikationsbeziehungen und Daten, die geschützt werden müssen, sind zu identifizieren. Entsprechend den Ergebnissen sind Maßnahmen zur Wahrung einer sicheren Kommunikation zu ergreifen. Für einzelne Komponenten (wie z. B. eine SPS) ist dies durch den Hersteller nicht möglich, da die Funktion und die darin verarbeiteten Daten vom Anwendungsfall abhängen und daher noch nicht bestimmbar sind.

Informationen zur Vorgehensweise bei der Schutzbedarfsanalyse können [BSI 100-3] entnommen werden. Entsprechend der Bewertungen sind Schutzmaßnahmen zu ergreifen.

M 6 Sicherstellung der Verfügbarkeit
Bei der Planung und Umsetzung sollte eine zuverlässige und robuste Möglichkeit für die Systemwiederherstellung berücksichtigt werden. Ein Beispiel ist das Vorgehen beim Austausch von einer defekten Komponente. Dazu gehören unter Umständen auch der Austausch von Authentisierungsdaten oder anderen kryptografischen Schlüsseln für den Zugriff auf Kommunikationsdienste.

Die Verfügbarkeit sollte auch dann sichergestellt sein, wenn sich Anforderungen durch die Security Policy im Krisenfall nicht direkt erfüllen lassen. So kann beispielsweise eine lokale Notabschaltung durch eine nicht berechtigte Person vorgenommen werden, muss aber entsprechend dokumentiert werden. Ebenso darf ein Ablaufen von Zertifikaten nicht zu einem Totalausfall führen oder muss durch organisatorische Maßnahmen (rechtzeitige Erneuerung) verhindert werden.

M 7 Security by Design
Bei dem Entwurf von Komponenten sollten so viele Securityfunktionen wie möglich verfügbar sein. Zur Reduzierung der damit einhergehenden Komplexität sollte sich dies jedoch auf die notwendigen Funktionen beschränken. Hersteller sollten anhand einer Risikoanalyse ermitteln, welche Bedrohungen für das Gerät bestehen. Grundlage sollten hierfür auch die Ergebnisse von Maßnahme M 5 sein.
Daraus resultiert eine Auswahl an Funktionen und ggf. Vorgaben, die es dem Integrator ermöglichen, eine sichere Maschine/Anlage zu bauen und mit dem Betreiber einen sicheren Betrieb gewährleisten zu können.

M 8 Verwendung von Open-Source-Software
Für viele Anwendungsgebiete gibt es Open-Source-Software. Es handelt sich hierbei um Software, deren Quelltext offengelegt ist. „Beim Einsatz der Freien Software sind dem BSI folgende technische Aspekte besonders wichtig:

1. Warnmeldungen über bei Sicherheitsprüfungen gefundene Fehler können veröffentlicht werden, weil es kein Non Disclosure Agreement gibt. Der Anwender kann so bei Sicherheitslücken schnell informiert werden und Gegenmaßnahmen ergreifen.
2. Die Prüfung von Software auf Sicherheitslücken sollte immer möglich sein. Beim Einsatz von Software kann dies ein Kriterium sein. Es steht Vertrauen versus Wissen.“ [BSI-FLOSS]

Sollen Open-Source-Komponenten verwendet werden, so sollte vor der Verwendung eine Überprüfung der Echtheit des Programms durchgeführt werden. Viele Open-Source-Projekte stellen entsprechende Informationen zur Verfügung. Erst nach erfolgreicher Prüfung sollte man die Komponente installieren/nutzen.

Bei der Integration von Open-Source-Komponenten in eigene kommerzielle Produkte sind auch die entsprechenden Lizenzbedingungen zu beachten. Einige Lizenzen erfordern beispielsweise die Offenlegung von Veränderungen am Source-Code.

Weitergehende Informationen können [GS-OSS] entnommen werden.

2.2. Reaktion auf Schwachstellen

Jeder Hersteller sollte auf die Entdeckung einer Schwachstelle in einer seiner Komponenten nach der Freigabe und dem Verkauf vorbereitet sein. Schwachstellen werden regelmäßig entdeckt, da fehlerhafte Implementierungen von Soft- und Hardware nicht ungewöhnlich sind. Je nach Art der Schwachstelle unterscheiden sich die Auswirkungen. Grundsätzlich sollte jedoch jede Art von Fehler beseitigt werden, soweit dies aus Kosten-/Nutzen-Aspekten vertretbar ist. Im Einzelfall kann auch die Empfehlung des Umstiegs auf eine neue Produktversion oder ein Nachfolgeprodukt geeignet sein. Hierbei sollte trotzdem ein Workaround vorgeschlagen werden, so dass Bestandskunden nicht völlig schutzlos sind.

M 9 Zentrale Kontaktmöglichkeit für Schwachstellenmeldungen
Es sollte eine zentrale und vertrauliche Möglichkeit zur Kontaktaufnahme beim Hersteller vorhanden sein, über die Schwachstellen gemeldet werden können. Die Kommunikation der Informationen zu der Schwachstelle an den Hersteller sollte zudem verschlüsselt erfolgen, da es sich um vertrauliche Daten handelt. Dazu kann ein öffentlicher Schlüssel in einem Verzeichnisdienst oder auf der Webseite hinterlegt werden.

Neben der Möglichkeit Kontakt mit dem Hersteller aufzunehmen, sollte ebenfalls eine zeitnahe Rückmeldung des Herstellers erfolgen. Diese sollte eine Bewertung der Schwachstelle und ggf. das weitere Vorgehen enthalten.

Tiefergehende Informationen sind in der Cybersicherheitsempfehlung „Handhabung von Schwachstellen“ [CS-E-Schwachstellen] enthalten.

M 10 Monitoring verwendeter Drittkomponenten
Der Umgang mit Schwachstellen beschränkt sich nicht nur auf Eigenentwicklungen. Er umfasst auch Produktbestandteile von Drittanbietern, die in eigenen Produkten zum Einsatz kommen. Hier gilt, dass die Relevanz und die Auswirkungen für die eigenen Komponenten geprüft werden müssen. Nach Vorliegen der Korrekturen für die Drittkomponenten sollte der Hersteller der Automatisierungskomponenten diese freigeben und ggf. Änderungen am eigenen Produkt vornehmen oder alternativ Workarounds definieren und kommunizieren.

M 11 Analyse von Schwachstellenmeldungen
Es sollten Ressourcen zur Verfügung stehen, um eingehende Meldungen zu Schwachstellen (aus den zwei zuvor genannten Maßnahmen) zu analysieren. Dazu gehört das Nachvollziehen der Schwachstelle und eine Einschätzung der Kritikalität dieser sowie die Prüfung von erarbeiteten Updates oder Workarounds.

Es sollte unternehmensintern eine Liste der vorhandenen Schwachstellen existieren.

M 12 Informationskanal zur Benachrichtigung von Kunden
Wurde eine Schwachstelle gefunden oder gemeldet, sollte eine Möglichkeit zur Behebung erarbeitet werden. Danach sollte eine Benachrichtigung der Kunden erfolgen. Dazu zählen Informationen, welche die betroffenen Funktionen und die Kritikalität des beseitigten Problems beschreiben. Die Kunden sollten so möglichst in die Lage versetzt werden, die eigene Betroffenheit bewerten zu können. Es sollten jedoch keine detaillierten Informationen bereitgestellt werden, die eine Ausnutzung der Schwachstelle begünstigen. Die Kommunikation sollte über vertrauenswürdige Kanäle erfolgen.

Bei einer sehr kritischen Schwachstelle oder falls die Behebung längere Zeit in Anspruch nimmt, sollte nach Möglichkeit zeitnah ein Workaround bereitgestellt werden, welcher die Ausnutzung einer Schwachstelle durch einen Angreifer verhindert oder zumindest erschwert.

2.3. Kryptografie

M 13 Auswahl der kryptografischen Verfahren
Der Bedarf und die Anforderungen (für z. B. Verschlüsselung oder Signaturen) sollten erhoben (vgl. [BSI-GS] M 2.162) und dokumentiert werden. Danach erfolgt eine für den Einsatzzweck geeignete Auswahl an kryptografischen Verfahren (vgl. [BSI-GS] M 2.164). Die Verfahren sollten die in den Komponenten zur Verfügung stehenden Ressourcen (z. B. eingeschränkte Rechenleistung) berücksichtigen. Bei der Auswahl sollte besonderes Augenmerk auf die Interoperabilität der Protokolle gelegt werden. Empfehlungen für kryptografische Algorithmen werden in [BSI-TR-02102] gegeben. Es sollte auf etablierte Bibliotheken zurückgegriffen werden. Auf eigene Implementierung sollte wegen der Komplexität verzichtet werden

Bei der Auswahl von geeigneten kryptografischen Verfahren sind auch die Auflagen der nationalen Außenwirtschaftsgesetze (Stichwort: Exportkontrolle) zu beachten. Hersteller sollten dieses Thema bereits in der Entwicklung entsprechend berücksichtigen.

M 14 Überwachung / Austausch von kryptografischen Verfahren
Neben der initialen Auswahl gilt es auch während des ganzen Lebenszyklus der Komponenten die Entwicklungen in diesem Bereich zu verfolgen. Es kann durchaus möglich sein, dass Schwächen in einem Verfahren bekannt werden und dieses daraufhin ausgetauscht werden sollte/muss. Dies sollte bereits bei der Entwicklung berücksichtigt werden.

M 15 Verwaltung kryptografischer Geheimnisse
Für den Betrieb ist es wichtig, dass

- die Speicherung der kryptografischen Geheimnisse in vor unberechtigtem Zugriff geschützter Weise erfolgt (ein Auslesen darf nicht möglich sein),
- ein Wechsel der Geheimnisse durch berechtigte Personen möglich sein sollte,

- ein Wechsel der Geräte möglich sein sollte, z. B. im Rahmen von automatisierten/ autarken Parametrier- und Konfigurationsvorgängen (wenn eine sichere Authentisierung und Autorisierung stattgefunden hat) und
- eine Generierung in einer geschützten und vertrauenswürdigen Umgebung erfolgt.

Für die Speicherung der kryptografischen Schlüssel kann z. B. ein Trusted Plattform Module (TPM) oder eine Smartcard zum Einsatz kommen. Hier ist zu beachten, dass in diesen der Speicherplatz für die Schlüssel begrenzt ist und die verfügbaren Algorithmen fest vorgegeben sind. Das bedeutet, es ist nicht ohne weiteres möglich, die Schlüssellänge zu verändern. Ein Austausch ist, wie z. B. im Falle eines TPM, nur durch einen Wechsel der Hardware möglich. Hier ist eine langfristige Planung notwendig.

Die Generierung von kryptografischen Geheimnissen ist eine sehr komplexe Materie. Hier sei auf [BSI-TR-02102] verwiesen.

2.4. Dokumentation

M 16 Interne Dokumentation
Bei der Entwicklung sollten Entscheidungen zum Design und zu Auswahlprozessen für Komponenten und Technologien dokumentiert werden. Dazu gehören auch Bedrohungs- und Risikoanalysen sowie umgesetzte Maßnahmen. Die genutzten Bibliotheken (auch Third-Party-Komponenten) und Versionen sollten ebenso erfasst werden wie auch die durchgeführten Tests und deren Ergebnisse. Diese Informationen sind für das Monitoring von Schwachstellenmeldungen notwendig (siehe Maßnahmen M 10).

Diese Dokumentation ist vor unberechtigtem Zugriff zu schützen.

M 17 Externe Dokumentation
Externe Dokumente sind von Herstellern für Integratoren, die Komponenten einsetzen, um Maschinen oder Anlagen zu bauen bzw. von Integratoren für Betreiber von Maschinen und Anlagen vorgesehen. Dies umfasst im wesentlichen Handbücher zu Komponenten, Komponentenserien, Maschinen oder Anlagen. Hier sollten alle Informationen enthalten sein, die für eine sichere Einrichtung und den sicheren Betrieb der Komponenten (im Sinne der Anlagenverfügbarkeit und -Security) notwendig sind. Ebenso sollen alle Informationen enthalten sein, die für Entscheidungen zur Systemarchitektur und zu implementierender Gesamtsysteme berücksichtigt werden müssen. Dazu gehören unter anderem:

- die verfügbaren Schnittstellen und Dienste (unterteilt in inaktiv und aktiv),
- die standardmäßig zugrunde gelegte Schutzbedarfsklassifizierung (hierfür kann [VDI2182-1] oder [BSI-100-2] herangezogen werden),
- Auflistung der Anforderungen an alle zum ordnungsgemäßen Funktionieren des Gesamtsystems notwendigen externen Ressourcen / Assets,
- aus Security-Sicht notwendige organisatorische und technische Maßnahmen,
- Dokumentation sämtlicher Produktfunktionen (insbesondere von Security-Konfigurationen, die vom Benutzer/Anwender nicht geändert werden können, weil sie beispielsweise fest codiert sind),
- Anleitungen für eine sichere Inbetriebnahme (z. B. Konfiguration von Diensten, Vergabe von Rechten, Authentisierungsmaßnahmen),
- Dokumentation der ggf. noch durchzuführenden Härtungsmaßnahmen (der Hersteller sollte auch eine Dokumentation zu den bereits durchgeführten und implementierten Härtungsmaßnahmen bereitstellen). Gerade bei der Inbetriebnahme eines neuen Gerätes könnte eine durchgeführte/implementierte und nicht dokumentierte Härtungsmaßnahme ggf. zu einem zusätzlichen Workaround führen um die benötigte Funktionalität zu realisieren. Zu den Härtungsmaßnahmen gehören Konfigurationen und Designaspekte.
- potentielle Risiken bei verschiedenen Konfigurationen sowie

- Hinweise zu weiteren Security-Maßnahmen.

2.5. Konfiguration

M 18 Minimalitäts-Prinzip hinsichtlich Berechtigungen
Benutzer sollten nur die Rechte erhalten, die für die Durchführung ihrer Aufgaben unbedingt benötigt werden (Minimalitäts-Prinzip, Least-Privilege). Dabei sollte insbesondere die Vergabe von Systemadministrator-Rechten vermieden werden. Nicht benötigte Accounts sollten deaktiviert oder wenn möglich entfernt werden (vgl. [BSI-GS] M 2.32).

Zu diesem Punkt zählen auch Systemaccounts. So ist es beispielsweise nicht notwendig, dass ein Webserver mit Administrator-Rechten betrieben wird. Es sollten daher alle Anwendungen und Dienste mit entsprechenden eingeschränkten Zugriffsrechten laufen.

M 19 Minimalitäts-Prinzip hinsichtlich der Systemeinstellungen
Komponenten in einem kritischen Umfeld sollten so konfiguriert sein, dass sie möglichst wenig Angriffspunkte bieten und lediglich notwendige Funktionen unterstützen. Daher sollten beispielsweise alle nicht benötigten Treiber entfernt und Entwicklungswerkzeuge (wie Compiler / Laufzeitumgebungen) nicht installiert werden. Dies reduziert nicht nur die möglicherweise durch einen Angreifer ausnutzbaren Schwachstellen, sondern vereinfacht auch die Pflege des Systems, da nur für die installierte Software Updates installiert werden müssen (vgl. [BSI-GS] M 4.95).

Wenn aus Sicht des Herstellers nicht bestimmt werden kann, welche Software-Komponenten ein Anwender benötigen wird, so ist dies im Rahmen einer konkreten Systemintegration durchzuführen.

Für optionale Dienste sollte für Betreiber oder Integratoren die Möglichkeit bestehen, diese zu aktivieren bzw. deaktivieren, wenn diese gebraucht bzw. nicht gebraucht werden.

M 20 Antivirensoftware
Die Möglichkeit zum Einsatz von Antivirensoftware auf den Komponenten sollte durch den Hersteller/Integrator geprüft werden. Bei dieser Prüfung sollte darauf geachtet werden, dass

- entsprechende Lösungen vorhanden sind,
- diese auf Rückwirkungsfreiheit getestet wurden (d. h. kein negativer Einfluss auf die ICS Systemleistung respektive das Systemverhalten),
- eine regelmäßige Aktualisierung der Virensignaturen erfolgen kann (d. h. die entsprechende Möglichkeit vorhanden ist, regelmäßige Updates einzuspielen) und
- eine Strategie konzipiert wurde, wie und ob ggf. im Laufe des Betriebs der Anlage ein Update der Antivirensoftware möglich ist (falls der Hersteller der Antivirensoftware keine Updates der Signaturen mehr bereitstellt).

Ein Ergebnis dieser Prüfung kann durchaus sein, dass der Einsatz einer Antivirensoftware nicht möglich oder nicht sinnvoll ist. Dies ist beispielsweise der Fall, wenn keine geeigneten Produkte zur Verfügung stehen oder die Versorgung mit aktuellen Virensignaturen aufgrund der Abschottung der Systeme nicht erfolgen kann oder eine Veränderung nicht zulässig ist. In diesen Fällen sollten andere Maßnahmen wie M 21, M 22 und M 23 umgesetzt werden.

Die notwendigen Informationen für die Konfiguration sollten durch den Hersteller/Integrator zur Verfügung gestellt werden.

Bei Embedded-Geräten sollte zumindest eine Veränderungserkennung erfolgen, um Manipulationen festzustellen und Prüfungen durchzuführen.

M 21 Application Whitelisting
Neben dem Schutz durch Antivirensoftware sollten Überwachungsmaßnahmen beim Start von Anwendungen vorgesehen werden.

Es sollte damit sichergestellt werden, dass nur freigegebene Versionen ausführbarer Dateien und keine eventuell unbefugt eingebrachten modifizierten Versionen oder Schadsoftware gestartet und ausgeführt werden können.

Änderungen an den Binärdaten der Anwendungen sowie an der Whitelist (Liste der ausführbaren Dateien) dürfen nur Administratoren gestattet sein (vgl. [BSI-GS] M 4.23).

Die Konfiguration kann bereits im Rahmen der Auslieferung durch den Hersteller erfolgen. Für die Installation und Konfiguration (z. B. bei Installation von anlagenspezifischer Software durch den Betreiber) der Application Whitelisting Software ist durch den Hersteller eine entsprechende Dokumentation zur Verfügung zu stellen.

M 22 Speicherschutz
Es sollten Mechanismen genutzt werden, um fehlerhafte Zugriffe (z. B. Buffer Overflow) auf den Arbeitsspeicher zu verhindern und das schadhafte Ausnutzen zu unterbinden. Dies gilt im wesentlichen beispielsweise für HMI, Leitsysteme und andere leistungsstärkere Systeme. In Embedded-Geräten stehen diese Funktionen in der Regel nicht zur Verfügung.

M 23 Schutz vor Prozessmanipulation
Es sollten Mechanismen genutzt werden, um die Manipulation von laufenden Prozessen, beispielsweise durch unbefugten dynamischen Austausch von Laufzeitbibliotheken oder Thread-Injection zu verhindern. Dies gilt im wesentlichen beispielsweise für HMI, Leitsysteme und andere leistungsstärkere Systeme. In Embedded-Ggeräten stehen diese Funktionen in der Regel nicht zur Verfügung.

M 24 Aktivierung von Schutzmechanismen bei Schnittstellen
Bei allen Schnittstellen sollten die gebotenen Sicherheitsmechanismen aktiviert werden. Dazu zählt beispielsweise die Nutzung von sicheren Verschlüsselungsverfahren sowie von Authentisierung. Dies gilt etwa für Bluetooth, WLAN (vgl. [BSI-GS] M 4.362) und Ethernet-basierte Feldbussysteme, sofern diese entsprechende Funktionen bereitstellen.

M 25 Hardware
Hardwareschnittstellen sollten, soweit sie nicht für den Betrieb notwendig sind, deaktiviert oder vor Missbrauch geschützt werden. Dazu zählen beispielsweise USB, JTAG oder SPI. Je nach Schutzbedarf kann die Deaktivierung z. B. über Betriebssystemfunktionen, über JTAG-Fuses oder das physische Blockieren der Schnittstellen mit Schlössern erfolgen. Eine weiteres Mittel ist die Verwendung von SecureJTAG.

Wenn ein sehr hoher Schutzbedarf vorliegt, sind auch Maßnahmen gegen eine physische Manipulation der Geräte (Tamper Resistance) denkbar. Grundsätzlich sollte der physikalische Zugang zur Hardware nur autorisiertem Personal erlaubt und möglich sein (z. B. durch Nutzung von abschließbaren Schränken/Racks und Zugangskontrollen).

2.6. Sicherheit der Anwendungen

2.6.1. Allgemeines

M 26 Validierung von Ein- und Ausgabedaten
Alle an die Komponente übergebenen Daten sollten vor einer Validierung als nicht vertrauenswürdig betrachtet werden. Bei der Validierung werden beispielsweise Längenbegrenzungen von Feldern und die Kodierung von Werten auf ihre Korrektheit geprüft. Wenn eine solche Prüfung unterbleibt, kann es bei der Verarbeitung unter Umständen zu Pufferüberläufen oder zu einer fehlerhaften Interpretation der Daten kommen. Pufferüberläufe gehören zu den größten Gefahren, da die Möglichkeit besteht, Schadcode einzuschleusen und zur Ausführung zu bringen.

Dies gilt insbesondere für Eingabefelder bei grafischen Benutzeroberflächen. Wenn eine solche Filterung der Ein- und Ausgabedaten technisch unmöglich ist (z. B. aufgrund sehr schneller zyklische Prozessdaten), kann darauf verzichtet werden. In diesen Fällen ist verstärkt auf die Robustheit Wert zu legen (vgl. Maßnahme M 4) und diese durch Tests zu prüfen.

Sofern entsprechende Informationen zur Verfügung stehen, kann zu der Validierung u. U. die Prüfung der Integrität und Authentizität zählen (vgl. [BSI GS M 4.393]).

Die Prüfung sollte grundsätzlich in der Komponente erfolgen. Zur Prüfung sollten folgende Schritte ergriffen werden:

- Prüfung auf die Länge und das Format von übergebenen Daten, Zeichen und Zeichenfolgen,
- Abgleich der Daten mit Listen, in denen nur erlaubte (Whitelist) oder verbotene (Blacklist) Zeichen/Zeichenfolgen enthalten sind. Das Whitelisting-Verfahren sollte präferiert werden.
- Es sollte eine Entfernung oder Umwandlung aller Zeichen vorgenommen werden, die als unsicher für einen möglichen Interpreter eingestuft werden. Dies kann beispielsweise ein Browser sein, der die Daten später erneut anzeigt oder eine Datenbank, in die Daten eingetragen werden. Dazu zählen z. B.
  - unerwartetes JavaScript und HTML zur Auslieferung an den Client (Web-Browser),
  - unerlaubt eingefügte SQL-Statements an die Datenbank (z. B. aus Eingaben in Formularfeldern)[1],
  - Befehle an das Betriebssystem (z. B. in manipulierten HTTP-Variablen).

M 27 Fehlerbehandlung
Als Fehler wird hier der Ausfall eines Teils einer Komponente (z. B. einer Netzwerkverbindung) oder die Unterbrechung der normalen Verarbeitung durch inkorrekte Daten (z. B. Eingangswert außerhalb des festgelegten Wertebereich durch Übertragungsfehler oder Manipulation) verstanden.

Tritt während des Betriebs einer Komponente solch ein Fehler auf, muss dieser so behandelt werden, dass ein konsistenter und stabiler Zustand der Komponente gewährleistet ist und damit der Schutz der Daten und ein sicherer Anlagenbetrieb aufrechterhalten wird, sofern dies möglich ist.

Eine Komponente ist in einem inkonsistenten Zustand, wenn sie aufgrund eines Fehlers in einen unerwarteten Zustand überführt wird und dadurch Daten unkontrolliert verarbeitet werden (z. B. keine Fehlermeldung bei erfolgloser Speicherung von Daten) (vgl. [BSI-GS] M 4.395).

Folgende Punkte sollten bei der Fehlerbehandlung berücksichtigt werden:

- keine Ausgabe von Informationen, die es einem Angreifer ermöglichen, Rückschlüsse auf den Aufbau der internen Struktur zu ziehen oder Schwachstellen leichter auszunutzen,
- Interne Protokollierung des Fehlerzustands mit Möglichkeit, über gesicherte Verfahren die Fehlerzustände und ihre Vorgeschichte (was zu ihnen geführt hat) auszulesen,
- geordneter Abbruch des Vorgangs nach Auftreten des Fehlers,
- Freigabe von zuvor reservierten Ressourcen nach Meldung / Auftreten des Fehlers,
- Berücksichtigung weiterer Aspekte wie z. B. das Zeitverhalten im Fehlerfall, um Auswirkungen auf den Steuerungsprozess zu vermeiden oder Rückschlüsse auf den Programmablauf/interne Informationen zu vermeiden,
- Sicherstellung des Anlagenbetriebs (z. B. keine unerwünschten Statusänderungen bei Anlagenkomponenten durchführen),
- Entwicklungstechnische Maßnahmen in Anwendungen, um fehlerhafte Speicherzugriffe (z. B. Buffer Overflow) zu verhindern und das schadhafte Ausnutzen zu unterbinden.

1 Hier sei auch noch auf [BSI GS M 2.363] zum Schutz gegen SQL-Injection hingewiesen.

2.6.2. Authentisierung & Autorisierung

Authentisierung bezeichnet den Nachweis eines Kommunikationspartners, dass er tatsächlich derjenige ist, der er vorgibt zu sein. Bei einer Autorisierung wird geprüft, ob eine Person, IT-Komponente oder Anwendung zur Durchführung einer bestimmten Aktion berechtigt ist.

M 28 Wechsel von Standard-Authentisierungsdaten
Der Betrieb eines Produktes mit unveränderten (d. h. den werksseitig voreingestellten) Sicherheitseinstellungen ist ein Sicherheitsrisiko. Hierzu gibt es eine Reihe von Lösungsmöglichkeiten, wie z. B.

- Erzwingen der Änderung bei Installation bzw. initialer Konfiguration.
- Hinweis in der Administrationsoberfläche, dass ein Standardpasswort/-schlüssel gesetzt ist.
- Hinweis in der Dokumentation – möglichst an herausragender Stelle – dass ein Standardpasswort/-schlüssel gesetzt ist und dieses dringend geändert werden muss.

Zusätzlich zu den genannten Umsetzungsmöglichkeiten gibt es weitere flankierende Maßnahmen wie z. B. das Empfehlen von Anforderungen an sichere Passwörter in der Dokumentation oder die technische Durchsetzung solcher Passwort-Richtlinien.

Es muss möglich sein, dass ein Betreiber alle Passwörter/Schlüssel austauschen kann, soweit dies nicht durch Anforderungen im Einsatzszenario und Randbedingungen ausgeschlossen ist.

M 29 Sicherer Austausch der Authentisierungsdaten
Bevor Authentisierungsdaten ausgetauscht werden, muss sich der Nutzer mit den bisherigen Daten authentisieren.

Solange ein Benutzer sich mit einem Passwort authentisieren muss, wird es vorkommen, dass er sein Passwort vergisst. Einerseits soll Benutzern in solch einer Situation schnell geholfen werden, damit sie wieder arbeiten können. Andererseits muss verhindert werden, dass sich Unberechtigte durch unzureichende Berechtigungsprüfungen Zugriff auf IT-Systeme verschaffen können.

Wichtig ist dabei, dass für das Zurücksetzen von Passwörtern flexible Richtlinien definiert werden. Einerseits sollte das Vorgehen dem Schutzbedarf des jeweiligen Passwortes entsprechen und andererseits sollten die Zugriffsanforderungen der Anwender berücksichtigt werden (vgl. [BSI-GS] M 2.402).

M 30 Sichere Speicherung von Authentisierungsdaten
Bei Passwörtern darf keine Speicherung im Klartext erfolgen. Es sollten sog. Salted-Hashes gespeichert werden, bei denen zu jedem Passwort ein zufälliger Wert ergänzt wird. Dieser Wert kann zusammen mit dem Passwort gespeichert werden. Der zufällige Wert muss nicht geheim gehalten werden. Er dient dazu, die Berechnung des gehashten Passwortes zu erschweren. Wenn dieser Wert nicht mit in das gehashte Passwort einbezogen wird, besteht die Möglichkeit sogenannte Rainbow-Tables im voraus zu berechnen. Diese enthalten ein Passwort und den zugehörigen Hashwert. Mit diesen muss lediglich der passende Hashwert gesucht werden und das ursprüngliche Passwort ist bekannt.

Bei alternativen Methoden ist ein Auslesen der Informationen zu verhindern (vgl. M 15). Im Falle von biometrischen Verfahren sei auf [BSI-BioKeyS] verwiesen.

M 31 Authentisierung bei Fernzugriff
Der Fernzugriff sollte nur nach erfolgreicher Authentisierung möglich sein. Für einen Fernzugriff sollten VPN-Techniken (z. B. IPSec, TLS) genutzt werden. Ergänzende, optionale Einschränkungen z. B. anhand von IP-Adressbereichen (sowohl auf Remote- als auch auf Anlagenseite) oder eine Verbindungstrennung nach einer bestimmten Zeit der Inaktivität sind sinnvoll.

M 32 Autorisation
Nach der erfolgreichen Authentisierung sollte in der Anwendung regelmäßig geprüft werden, ob der Nutzer auch die Rechte hat, auf die angeforderten Ressourcen/Daten zuzugreifen. Dies ist insbesondere erforderlich, wenn unterschiedliche Rollen vorhanden sind.

2.6.3. Betrieb und Überwachung

M 33 Protokollierung
Sicherheitsrelevante Ereignisse (z. B. Zugriffe auf Ressourcen, Authentisierungsversuche) sollten unter Nennung von Zeitpunkt, dem betroffenen Dienst/System/Anwendung, dem genutzten Benutzeraccount und der Ursache protokolliert werden, damit im Stör- oder Fehlerfall oder nach Angriffsversuchen die Protokolldaten zur Ursachenfindung herangezogen werden können (vgl. [BSI-GS] M 4.397). Dabei sind insbesondere z. B. folgende Vorkommnisse von Interesse (vgl. [BSI-GS] M 5.9):

- falsche Passworteingabe für eine Benutzer-Kennung,
- Versuche von unbefugten (lokalen wie Fern-)Zugriffen,
- Daten zur Ressourcenauslastung und -überlastung,
- Anschließen, Entfernen und Austausch von Wechseldatenträgern,
- Konfigurationsänderungen.

Die Protokolldaten sollten an ein zentrales System übermittelt werden. Um unbefugte Veränderungen von Protokolldaten aufdecken zu können, sind diese Daten (z. B. durch Prüfsummen) zu schützen. Bei der Übermittlung an ein zentrales System ist die zusätzlich entstehende Last auf dem Netzwerk zu berücksichtigen. Überlastungen sollten in jedem Fall vermieden werden.

In den Protokollierungsdaten dürfen keine sensiblen und vertraulichen Daten, wie z. B. Passwörter, enthalten sein.

Es sollte die Möglichkeit bestehen, die Komponente an ein zentrales Logging und Monitoring anzubinden.

2.7. Updates & Wiederherstellung

M 34 Updates
Es sollte eine Möglichkeit bestehen, sich über neue Updates und Workarounds informieren zu lassen, etwa wenn kurzfristig Schwachstellen in genutzten Komponenten bekannt werden (vgl. M 12).

Bei der Bereitstellung von Updates sollte darauf geachtet werden, dass

- diese nur bei erfolgreicher Prüfung auf ihre Authentizität und Integrität (z. B. mittels Hashwert) hin akzeptiert werden.
- vom Hersteller dokumentiert ist, welche Auswirkungen das Update auf die Sicherheitsfunktionen hat. Dazu zählt, ob bzw. welche Maßnahmen ggf. angepasst werden müssen (z. B. bei Einsatz von Application Whitelisting, siehe Maßnahme M 21).
- ob bzw. mit welchen Nichtverfügbarkeiten zu rechnen ist.
- zwischen kritischen und optionalen Updates unterschieden wird.
- die Installationsbedingungen klar dokumentiert sind (z. B. welche Vorversionen müssen installiert sein).
- Informationen zu neuen oder veränderten oder eingeschränkten Funktionalitäten enthalten sind.
- der Betreiber über geänderte Voreinstellungen oder zurückgesetzte Konfigurationen Kenntnis erhält.

M 35 Sicherung und Wiederherstellung
Es sollte die Möglichkeit bestehen, die Konfiguration und die vorhandenen Daten zu sichern und wiederherzustellen. Sicherungen sollten vor nachträglichen Veränderungen (z. B. durch digitale Signaturen) sowie unbefugtem Zugriff geschützt werden (z. B. verschlüsselte Ablage).

2.8. Protokollspezifika

M 36 Verwendung sicherer IT- bzw. Standardprotokolle
Auf Anwendungsebene sollten sichere Varianten eines Protokolls sowie flankierende Mechanismen zur Verschlüsselung sowie der Integritäts- und Authentizitätsprüfung zum Einsatz kommen. Beispiele hierfür sind die Verwendung von HTTPS anstatt HTTP oder FTPS statt FTP.

Für weitere unterstützende Dienste, wie beispielsweise DNS, existieren ebenfalls sichere Varianten – DNSSEC – bei denen jedoch aufgrund der bisher geringen Verbreitung nur in Einzelfällen ein Einsatz möglich ist. Hier muss entsprechend des bestehenden Risikos entschieden werden, ob diese genutzt werden.

Gegebenenfalls sind Vorkehrungen zu treffen, dass beispielsweise über einen Schlüsselrückruf keine Beeinträchtigung von anderen Diensten (z. B. DNS) und der Verfügbarkeit von Infrastrukturen stattfindet.

M 37 Verwendung von aktuellen IT- bzw. Standardprotokoll-Versionen
Es sollten die jeweils aktuellen Versionen der Protokolle zum Einsatz kommen. Bei der sicheren Kommunikation über TCP sollte beispielsweise TLS 1.2[2] anstelle älterer Versionen genutzt werden. Neben der reinen Protokollversion sollte auch bei der Wahl der Ciphersuites auf als sicher anerkannte gesetzt werden.

Ähnliches gilt für die Verwendung von SNMP (Simple Network Management Protocol). Hier sollte SNMPv3 genutzt werden. Dort wurden Funktionen wie Verschlüsselung und eine verbesserte Authentisierung eingeführt. Zu beachten ist, dass auch bei SNMPv3 gewisse Restrisiken verbleiben. Insbesondere Brute Force- und Wörterbuch-Angriffe auf die Authentisierung sind möglich. Daher ist es sinnvoll, einen geeigneten Detektionsmechanismus für solche Angriffe (unabhängig vom eingesetzten Protokoll) zu integrieren. Zusätzlich sollte ein schreibender Zugriff nur wenn nötig implementiert werden bzw. dieser sollte deaktivierbar sein.

Werden aus Gründen der Kompatibilität ältere Protokollversionen unterstützt, sollten diese erst nach expliziter Aktivierung durch den Betreiber genutzt werden können bzw. sollten deaktivierbar sein. Der Hersteller hat in der Dokumentation ausdrücklich auf ältere/unsichere Protokollversionen und die zu erwartenden Funktionseinschränkungen bei Nicht- oder Deaktivierung dieser Protokolle (durch den Betreiber) hinzuweisen.

M 38 XML
Der eingesetzte Parser für XML-Daten sollte möglichst restriktiv konfiguriert sein. Dazu zählt beispielsweise, dass Verweise auf externe Ressourcen nicht aufgelöst und abgefragt werden oder dass Systembefehle nicht ausgeführt werden.

Vor der Verarbeitung sollten die XML-Daten auf anwendungsspezifische Angriffsmöglichkeiten geprüft (z. B. zu tiefe Verschachtelung, zu viele Elemente) werden.

2.9. Integration

Integratoren sollten Security bereits bei der Planung als elementaren Aspekt berücksichtigen. Dazu gehören unter anderem:

- architekturelle Aspekte,
- sorgfältige und gewissenhafte Konfiguration und
- Hinweise für die Integration der Maschine/Anlage in ein ggf. bereits vorhandenes Sicherheitsmanagement des Betreibers (z. B. auch organisatorische oder den physikalischen Aufbau betreffend).

Zusätzlich zu den Maßnahmen aus den zuvor genannten Kapiteln sollten noch die nachfolgenden Maßnahmen berücksichtigt werden.

2 Stand 10/2014

Der Integrator nutzt die in den Komponenten enthaltenen Security-Funktionen um eine mehrstufige Absicherung der Maschine/Anlage durchzuführen. Dabei sollte jede Komponente, entsprechend den ihr zur Verfügung stehenden Funktionen abgesichert sein (siehe M 39). Um die einzelnen Komponenten, Maschinen- oder Anlageteile untereinander abzusichern, sollte eine Unterteilung in Schutzzonen erfolgen (siehe M 40 und M 41). Auf diese Weise reihen sich verschiedene Schutzmechanismen aneinander. Dieser Ansatz wird Defense-in-Depth genannt. Mit jeder Schicht werden weitere Security-Funktionen ergänzt.

Die Ergänzung von Security-Funktionen in äußeren Schichten bedeutet nicht, dass die inneren Teile der Maschine gänzlich auf das Thema Security verzichten können. Es sind vielmehr zusätzliche Hürden, die ein Angreifer überwinden muss.

M 39 Aktivierung der Security-Funktionen
Der Integrator sollte die von den Komponenten angebotenen Security-Funktionen der Komponenten standardmäßig aktivieren und nutzen, sofern er bei einer zuvor erfolgten Betrachtung festgestellt hat, dass diese aktuell und für den Anwendungsfall angemessen sind.

M 40 Vertikale Segmentierung
In einer Anlage sind die unterschiedlichen Bereiche wie beispielsweise Produktion und Office-IT voneinander durch Firewalls oder andere, filternd wirkende oder überwachende IT-Systeme zu trennen. Eine Segmentierung kann bei Bedarf auch durch ein Perimeter-Netzwerk (DMZ) erfolgen. Damit werden unberechtigte Zugriffe zwischen den Teilnetzen erschwert bzw. verhindert.

M 41 Horizontale Segmentierung
Verschiedene Anlagen-/Maschinenteile sollten durch eine Firewall voneinander segmentiert werden. Hierdurch wird verhindert, dass sich beispielsweise Schadsoftware ungehindert in der gesamten Anlage ausbreiten kann. Mögliche negative Beeinflussungen auf Komponenten und Anlagenbetrieb wirken somit nur auf begrenzte Teile der Anlage ein. Dies gilt auch für Bereiche mit unterschiedlichem Schutzbedarf.

M 42 Einbindung in Informationssicherheitsmanagementsystem (ISMS)
Der Integrator sollte zur Einbindung einer Maschine, Anlage oder von Anlagenteilen in ein vorhandenes Informationssicherheitsmanagementsystem (z. B. nach ISO 27001, [ISOIEC-62443-2-1] oder [BSI-ICS]) die diese betreffenden Informationen bereitstellen. Falls Drittanbieter-Komponenten bisher nicht vorgesehen sind, sollte dem Betreiber dennoch für zukünftig geplante Anbindungen die Einhaltung von entsprechenden Security-Vorgaben empfohlen werden.

Hierzu gehören im Fall von Fernwartungsinstallationen auch Details dazu, welche Tätigkeiten möglich bzw. erforderlich sind und welche Informationen übertragen werden. Es muss einen geregelten Prozess für die Fernwartung zwischen Betreiber und dem Nutzer der Fernwartung erstellt werden, indem Zugriffsrechte, Protokollierung und notwendige Freigabe- bzw. Genehmigungsverfahren klar und nachvollziehbar geregelt sind.

3. Prüfung der Komponenten

3.1. Einführung zum Thema Security-Tests für Komponenten

Durch Sicherheitstests sollen Bedrohungen so früh wie möglich aufgedeckt werden, um diesen noch vor Auslieferung bzw. Inbetriebnahme mit wirksamen Maßnahmen entgegenwirken zu können.

Als positiver Nebeneffekt ergibt sich aus Security-Tests eine geringere Fehlerhäufigkeit in Produkten und damit eine Kostenreduktion durch Verringerung des Nachbesserungsbedarfs, eine Bereicherung des Testprozesses um zusätzliche Negativtests und nachhaltige Lernprozesse für die Entwicklung.

Die Testdurchführung sollte

- bei Herstellern und Integratoren, während der Neu- oder Weiterentwicklung von Komponenten und Anlagen oder Factory Acceptance Tests
- bei Betreibern, im Rahmen von Site Acceptance Tests

mittels strukturierter und reproduzierbarer Prüfungen erfolgen.

Die Literatur bietet hierzu verschiedene Vorgehensmodelle für Security-Tests an. An dieser Stelle sei auf einige hingewiesen:

- Open Source Security Testing Methodolodgy Manual [OSSTMM] behandelt unterschiedliche Arten von Security-Tests,
- Embedded Device Security Assurance [ISA-EDSA] zielt auf den Test von Embedded Devices ab und
- OWASP Testing Guide [OWASP-2] ist auf Webanwendungen fokussiert.

Die im Weiteren beschriebenen Security-Tests wurden aus praktischen Erfahrungen im regulären Testmanagement und bei Penetrationstests zusammengestellt. Die in Kapitel 4 dargestellten Fragestellungen sollten während der Entwicklung berücksichtigt und am Ende der Entwicklung auf ihre Wirksamkeit geprüft werden.

3.1.1. Grenzen

Die beschriebenen Hinweise helfen die Security in Komponenten zu verbessern. Sie können jedoch nicht sicherstellen, dass alle Schwachstellen gefunden werden. Dies ist aufgrund der Komplexität der Komponenten und begrenzter Ressourcen nicht möglich. Außerdem beziehen sich die Hinweise auf bereits bekannte Schwachstellen. Bisher unveröffentlichte Angriffswege sind daher unberücksichtigt und können nicht direkt geprüft werden.

3.1.2. Testteam, Testfälle, Testaufbau und Testprozess

Um das Testziel zu erreichen, sollte sich die Bildung des Testteams sowie die Ausgestaltung des Testprozesses an gängigen und bewährten Standards (z. B. ISO/IEC/IEEE 29119 "Software Testing") orientieren.

Es wird insbesondere empfohlen,

- das Testteam nicht mit Mitarbeitern aus dem jeweiligen Entwicklerteam zu besetzen, um eine neutrale Bewertung des Systems zu garantieren.
- Testfälle möglichst transparent und reproduzierbar zu gestalten.
- den Testaufbau möglichst an praktischen Szenarien zu orientieren, um belastbare Testergebnisse zu erreichen.

- den Testprozess so zu gestalten, dass transparente Ergebnisse zur anschließenden Fehlerkorrektur und Bildung wirksamer Maßnahmen geliefert werden.
- nach erfolgter Fehlerkorrektur und Etablierung wirksamer Maßnahmen die Lösung erneut im Testprozess zu prüfen.

3.1.3. Lernprozess und nachhaltige Qualitätssteigerung

Aus dem Feedback des Testteams resultiert eine zunehmende Fachkenntnis und Sensibilität im Entwicklungsteam für die Vermeidung von Fehlern oder architekturellen Schwachstellen.

3.2. Empfehlung zur Testdurchführung

In diesem Kapitel werden Empfehlungen zur Durchführung von Security-Tests beschrieben. Die Empfehlungen sind weitestgehend kompatibel zu Best Practices aus dem Bereich des Testmanagements (z. B. ISO/IEC/IEEE 29119 "Software Testing") und lassen sich daher in ein bereits vorhandenes Testmanagement integrieren.

3.2.1. Erstellung eines Testplans

Wie jedes gut strukturierte Projekt sind Security-Tests zu planen. In der Projektvorbereitung sollten Aufwände und Zeitrahmen geschätzt und im Projektverlauf kontrolliert und gegebenenfalls angepasst werden.

Bei der Auswahl der Testwerkzeuge sollte die Funktion und der Anwendungsbereich geprüft werden, um mittels einer geeigneten Auswahl brauchbare Ergebnisse zu erhalten. Mit der Ausweitung der Security-Tests sollte eine verstärkte Automatisierung erfolgen. Dies reduziert die Aufwände. Außerdem wird hierdurch die Reproduzierbarkeit der Tests verbessert und aussagekräftigere Testergebnisse erzielt.

3.2.2. Testvorbereitung

Zur Spezifikation von Testfällen und der Bewertung von Testergebnissen benötigt das Testteam ein ausgereiftes technisches und funktionales Verständnis des zu prüfenden Systems sowie ein Verständnis von möglichen Schwachstellen und Security-Mechanismen. Eine angemessene Einarbeitungsphase ist für das Testteam unabdingbar und bildet die Basis für aussagekräftige und belastbare Testergebnisse.

Zur effizienten Ausrichtung der anstehenden Tests hat sich die folgende Vorgehensweise bewährt:

1. Übersicht verschaffen
   Das Testteam verschafft sich eine Übersicht über das zu testende System und zu erwartende Einsatzszenarien auf einer funktionalen Ebene. Dazu haben sich sowohl Produktbeschreibungen, Zusammentreffen mit Systemarchitekten, Vorträge als auch Fachartikel bewährt. Ebenfalls hilfreich können Anforderungsspezifikationen sein, die neben den reinen Produktfeatures auch die grundlegenden Anforderungen enthalten.
2. Dokumentation erheben und sichten
   Nachdem das Testteam eine Übersicht erlangen konnte, werden sämtliche verfügbaren Dokumentationen gesammelt und gesichtet. Es wird empfohlen, bereits während der Sichtung offene Fragen, identifizierte Widersprüche und Anregungen für Testszenarien unmissverständlich und dennoch stichpunktartig zu dokumentieren.
3. Offene Fragen und Widersprüche klären
   Eventuell offene Fragen und Widersprüche mit Testrelevanz sind aufzulösen. Es wird von der verbreiteten Vorgehensweise abgeraten, offene Punkte während der Testphase am Testobjekt klären zu wollen. Dies könnte zu unwirtschaftlichen Aufwänden, unnötigen Nacharbeiten in

Testfallspezifikationen und insbesondere Fehlinterpretationen mit wertlosen Testergebnissen führen.

4. Erste Einschätzung zu möglichen Schwachstellen
   Auf Basis der gesammelten Informationen kann eine erste Einschätzung zu möglichen Schwachstellen gegeben werden. Dabei handelt es sich um Aspekte, die in den ersten Schritten der Security-Tests genauer betrachtet werden sollten. Eine reine Beschränkung auf die hier identifizierten Themen sollte jedoch nicht stattfinden.

3.2.3. Erstellung von Testfälle und Testszenarien

In Vorbereitung auf die aktive Testdurchführung werden Testfälle erarbeitet. Mit jedem Testfall soll geprüft werden, ob das zu testende System gegenüber einer Bedrohung resistent ist (Negativtest) oder ergriffene Maßnahmen wirksam einer Bedrohung entgegenwirken (funktionaler Test). Bei logischen oder chronologischen Zusammenhängen einzelner Testfälle lassen sich diese zu einem Testszenario zusammenfassen.

Testfälle und Testszenarien sollten die folgenden Bereiche abdecken:

- Nachweis einer definierten Robustheit
- Erkennen von Schwachstellen (z. B. auch in Drittkomponenten, die eingesetzt werden)
- Zielgerichtete systemspezifische Security-Tests

Aus den bekannten Details und vorliegenden Dokumenten werden die Testfälle und Testszenarien abgeleitet. Eine angemessene schriftliche Dokumentation der jeweils durchzuführenden Tests sowie der zu erwartenden Ergebnisse ist unabdingbar. Damit werden sowohl während der Testdurchführung Fehlinterpretationen durch testende Personen verringert als auch später die Reproduktion und Bewertung von Befunden ermöglicht. Erfahrungsgemäß ist die Belastbarkeit und Qualität der Testergebnisse unmittelbar an Form und Qualität der erstellten Testszenarien gekoppelt.

Als Testszenarien können folgende Hinweise genutzt werden:

- Anregungen für Testfragen aus Kapitel 4 dieses Dokuments
- Welche Szenarien würden im produktiven Betrieb die Schutzziele gefährden?
- Wie könnte ein Angreifer technisch und/oder physisch das System und damit verbundene Daten und Prozesse schädigen?
- Welche Security-relevanten Auswirkungen könnten Konfigurationseinstellungen haben?
- Welche systemrelevanten Auswirkungen könnten Security-Konfigurationseinstellungen haben?
- Welche Security-relevanten Szenarien sind in der Vergangenheit bereits eingetreten?
- Welche Auswirkungen können Fehlbedienungen haben?

Bei der Erarbeitung von Security-relevanten Testszenarien sollte Folgendes beachtet werden:

- Bei der Erstellung der Testszenarien sollte davon ausgegangen werden, dass ein potenzieller Angreifer über Insiderwissen verfügt.
- Testszenarien können kombinierte Schritte beinhalten - zu komplexe Testszenarien sollten jedoch vermieden werden. Stattdessen sind Testszenarien mit einem definierten Ausgangszustand des Systems meist zielgerichteter.
- Einzelkomponenten sollten unter dem Aspekt der Fehlerfreiheit und Security getestet werden und im ersten Schritt keine zusätzlichen Schutzmechanismen (z.B. externe Firewall) berücksichtigen. Erst im zweiten Schritt ist eine Einbeziehung dieser Security-Mechanismen sinnvoll.

- Es sollten Schwachstellen-Scanner eingesetzt werden, die automatisiert auf bereits bekannte Schwachstellen in Soft- oder Hardware prüfen. Auf diese Weise kann sichergestellt werden, dass bekannte Schwachstellen in neuen Systemen vermieden werden.
- Die prüfende Person sollte im Befund einen Freitext angeben können. Anhand dieser Anmerkungen kann in der Nachbereitung der Bedarf für zusätzliche Testszenarien identifiziert werden. Eine Einschätzung zur Testabdeckung und -tiefe sollte ebenfalls Bestandteil sein.

3.3. Generelle Angriffsvektoren

Unabhängig von der konkreten Implementierung und dem Anwendungsprotokoll (z. B. Modbus, HTTPS) lassen sich mögliche Angriffsvektoren über externe Schnittstellen auf ICS grundsätzlich in sechs Bereiche untergliedern. Es handelt sich um die Reaktion des Systems auf:

1. Zufällige Eingangsdaten
2. Änderungen in den Übertragungsparametern (Geschwindigkeit der Übertragung, Typ und Größe der Datenpakete etc.)
3. Alle für das jeweilige Protokoll spezifizierten Nachrichten-Typen
4. Eingaben, die zwar die entsprechende Protokoll-Spezifikation inhaltlich erfüllen, jedoch dabei die Struktur der Nachrichten auf nicht standardkonforme Werte ändern (Längen der Variablen, Reihenfolge der Parameter etc.)
5. Eingaben, die zwar die Protokoll-Spezifikation strukturell erfüllen, jedoch den Inhalt von Variablen auf nicht standardkonforme Werte ändern (Gültigkeitsbereich der Variablen, negative Werte, fehlerhafte Metadaten der Variablen etc.)
6. Wiedereinspielen von aufgezeichneten Nachrichten

Jeder dieser sechs Bereiche ist ein potenzieller Angriffsvektor, denn bei einer fehlerhaften Verarbeitung kann es zu einer Kompromittierung des Systems oder zu Einschränkungen der Verfügbarkeit bzw. Verarbeitungsgeschwindigkeit kommen.

Angreifer zielen dabei auf in der Spezifikation fehlende Beschreibungen für Tests bzw. Überprüfungen von Parametern oder fehlerhafte Implementierungen ab. Insbesondere die Gefährdung der Stabilität einer Komponente durch eine unzureichende oder fehlende Validierung eines Parameters stellt ein erhöhtes Risiko dar. Es stellt sich häufig heraus, dass Entwickler diesem Bereich zu wenig Aufmerksamkeit schenken, sodass auf solch unerwarteten Input undefiniert reagiert wird. Dies kann mit stark verzögerter Verarbeitungsgeschwindigkeit aufgrund von Vollauslastung der internen Ressourcen, einem Neustart oder aber auch dem kompletten „Einfrieren“ des Gerätes verbunden sein, sodass ein manuelles Zurücksetzen des Gerätes notwendig wird.

3.3.1. Testszenarien zur Prüfung auf Robustheit

Mit Robustheitsprüfungen soll das Verhalten der zu testenden Komponente außerhalb eines strukturierten Laborbetriebs simuliert und beobachtet werden. Es handelt sich hierbei um Wechselwirkungen oder externe Einflüsse. So sollte beispielsweise eine Komponente nicht durch einzelne oder sehr viele ICMP-Nachrichten gestört werden.

Zur weiteren Anregung für geeignete Testszenarien mit dem Ziel der Robustheitsprüfung werden nachfolgend mehrere Aspekte beispielhaft aufgeführt. Einige Testszenarien mögen trivial erscheinen, werden aber häufig nicht berücksichtigt. Hier kommt es auf das Einsatzgebiet und die Möglichkeiten, die die verfügbaren Schnittstellen bieten, an. Nicht alle nachfolgenden Fragen sind daher überall relevant.

- Verhält sich das System auch bei häufigen und in schneller Abfolge wechselnden Funktionsaufrufen / Aktivitäten weiterhin stabil?

- Gefährden Fehlbedienungen zu keiner Zeit die Stabilität des Systems?
- Werden Wertebereiche stets korrekt geprüft und Fehleingaben abgefangen? Werden die Grenzbereiche explizit getestet?
- Läuft das System nach Betriebsunterbrechung durch Stromausfall weiterhin stabil?
- Werden Prozesse konsistent fortgesetzt, nachdem diese durch Stromausfall unterbrochen wurden?
- Ist die Stabilität bei Netzwerkempfang von Daten gängiger Protokolle (wie z. B. ICMP mittels Ping, SNMP, Broadcasts, usw.) gewährleistet?
- Verhält sich das System stabil und integer gegenüber unerwartet angeschlossenen und entfernten Devices / Schnittstellen (wie z. B. USB Geräte, Netzwerkkabel, Routing, usw.)?

3.3.2. Prüfung auf bekannte Schwachstellen

Durch Prüfungen auf bekannte und veröffentliche Schwachstellen soll, soweit möglich, eine unnötige Anfälligkeit und die damit verbundenen Risiken für den Testgegenstand frühzeitig erkannt werden. Dies trifft im wesentlichen auf die Verwendung von Drittkomponenten zu.

Die Prüfungen auf verbreitete Schwachstellen sollen vorsätzliche und möglicherweise schädliche Handlungen Dritter verhindern, da diese meist mit einem durchschnittlichen Aufwand (Zeit und Wissen) ausgenutzt werden können. Verbreitete Schwachstellen werden von Angreifern häufig durch automatisierte Schwachstellen-Scans ermittelt. Daher wird empfohlen, vergleichbare Werkzeuge für die Prüfung einzusetzen.

Zur weiteren Anregung für geeignete Testszenarien mit dem Ziel der Schwachstellenprüfung werden nachfolgend mehrere Prüffragen beispielhaft aufgeführt:

- Werden durch verbreitete und leistungsfähige Schwachstellen-Scanner und Manager (z. B. das freie OpenVAS[3]) Schwachstellen gefunden?
- Ist das System bei Auslieferung frei von (bekannter) Schadsoftware?
- Stehen Meldungen von Herstellern, CVE, CERTs oder anderen Quellen über bekannte Schwachstellen für sämtliche im Testobjekt genutzten Komponenten (Hard-/Software) in der eingesetzten Version und Betriebsart zur Verfügung?

3.3.3. Testansatz für Geräte mit Schnittstellen

Beim Test von Komponenten mit Schnittstellen sollten zwei übergeordnete Testansätze verfolgt werden:

1. Prüfung des Kommunikationsprotokolls
   In diesem Teil wird die Robustheit und Security der Umsetzung der Kommunikationsprotokolle geprüft. Es geht hierbei nur um Aspekte der Übertragung.
2. Prüfung der Daten und Verarbeitungslogik
   Der zweite Teil beschäftigt sich mit den Inhaltsdaten und deren Verarbeitung. Hierbei geht es um die sichere Verarbeitung der Daten (z. B. bei Überschreitung von Wertebereichen).

Im Falle, dass das zu testende Gerät mehrere Schnittstellen und Protokolle unterstützt, sind alle Schnittstellen und Protokolle zu prüfen. Wenn technisch nicht ausgeschlossen werden kann, dass die Schnittstellen und Protokolle eines Gerätes auch parallel genutzt werden könnten, so sind diese Tests auch parallel durchzuführen, um die Separierung der einzelnen Zugriffsarten auf das Gerät ebenfalls zu testen bzw. die gegebenenfalls maximale Ressourcenausnutzung des Gerätes in so einem Falle (Peak- bzw. Worst-Case-Last) zu berücksichtigen.

3 http://www.openvas.org

Das nachfolgende Beispiel soll den Testansatz verdeutlichen:

Beispiel: Webanwendung zur Administration
Hierbei sollten sowohl die Implementierung für den TCP- und den HTTPS-Stack geprüft werden. Dabei geht es um die korrekte Verwendung der Zertifikate und privaten Schlüssel für die Verbindungen und die Reaktion beispielsweise auf abgelaufene oder nicht vertrauenswürdige Zertifikate. Die eigentlichen Daten und deren Verarbeitung in der Webanwendung fallen unter den zweiten Prüfaspekt.

3.3.4. Zielgerichtete Security-Tests

Mittels der zuvor beschriebenen Tests sollten bereits wesentliche Schwachstellen einer Komponente aufgedeckt worden sein. Es besteht zusätzlich die Möglichkeit nach spezifischen Schwachstellen einer Komponente zu suchen, die sich aus den Eigenschaften einer Kombination der Soft- und Hardware ergeben.

Diese Art der Prüfungen bezieht sich auf vorsätzliche und potenziell schädliche Handlungen Dritter mit überdurchschnittlichen Wissen und zielgerichteter Motivation. Diese Art von Test sollte erst ins Auge gefasst werden, wenn ein Testmanagement etabliert ist.

3.3.5. Planung und Bereitstellung

Mit fortschreitender Erstellung der Testszenarien werden die zu berücksichtigenden Rahmenbedingungen der Testdurchführung deutlich. Abhängig von den Testszenarien und dem zu testenden System kann eine Unterteilung in Teilkomponenten mit abschließenden Tests des Gesamtsystems notwendig werden.

Der Aufbau der Testumgebung sollte dokumentiert werden, um die Nachvollziehbarkeit für folgende Prüfungen zu gewährleisten.

- Sollte das zu testende System bereits mit Security-Maßnahmen (Daten-Diode, Firewall, NIDS, usw.) ausgestattet sein, ist es ratsam, die geplanten Testszenarien sowohl mit als auch ohne aktivierte Security-Maßnahmen durchzuführen. Diese Vorgehensweise garantiert, dass in die Testergebnisse sowohl mögliche Bedrohungen für einzelne Komponenten und / oder das Gesamtsystem als auch die Wirksamkeit der angedachten Security-Maßnahmen eingehen.

3.4. Durchführung

3.4.1. Bereitschaft des Testobjekts

Vor jedem Test ist das Testobjekt in den für den Testfall definierten Zustand zu versetzen. Dazu gehören beispielsweise Konfigurationen, Verkabelungen oder sonstige Betriebszustände. Die korrekte Ausgangssituation ist für eine erfolgreich Durchführung von grundsätzlicher Bedeutung, da ansonsten falsche Testergebnisse geliefert oder deren Reproduzierbarkeit eingeschränkt werden können.

3.4.2. Monitoring

Im Fokus vieler Security-Tests steht die Reaktion der zu testenden Geräte, insbesondere ihre Stabilität, sodass empfohlen wird, diese während des gesamten Prüfvorgehens zu überwachen. Aufgrund der unterschiedlichen Schnittstellen, deren Implementierung sowie der Möglichkeiten zur Überwachung der Komponenten können an dieser Stelle nur allgemeine Vorschläge unterbreitet werden. Es ist zu beachten, dass Schnittstellen (z. B. ein hardware-basierter Kommunikationsstack) ggf. unabhängig von der zu testenden Funktion realisiert sein können. Solche Wechselwirkungen sind zu berücksichtigen.

- Nutzung von Netzwerkverbindungen durch permanente Messung der Latenzzeiten (z. B. ICMP-Echo-Ping Messung). Hier ist die zuvor genannte ggf. unabhängige Realisierung des Kommunikationsstack zu beachten.

- Bei direkter Verbindung (z. B. USB oder serielles Kabel) Nutzung eines Protokoll-Monitors (Hardware oder Software), um den ausgehenden Verkehr bzw. dessen Ausbleiben zu detektieren.
- Im Gerät implementierte Anzeigen/Alarme, die eine Überlast bzw. einen Neustart des Gerätes sichtbar machen. Entwicklungsbegleitend oder zur genaueren Untersuchung einer Schwachstelle kann, sofern vorhanden, auf Debug-Schnittstellen oder Tools zum Profiling oder Monitoring von Binärprogrammen bzw. Quellcode zurückgegriffen werden. Hier sollte auf mögliche Seiteneffekte geachtet werden.
- Optische Überwachung mittels Kamera bei Geräten mit sichtbaren Anzeigen.

3.4.3. Testdurchführung

Die Testdurchführung erfolgt gemäß der Testplanung in vorgesehener Reihenfolge und mit den vorgesehenen Ressourcen.

Während der Testphase haben sich zeitnahe Rückmeldungen der testenden Person mit Testergebnissen und Anmerkungen an das Entwicklungsteam bewährt. Dies ermöglicht es, frühzeitig Missverständnisse und Fehler in der Beschreibung von Testfällen und Testszenarien zu identifizieren und Anpassungen vorzunehmen. Zusätzliche Testfälle und Szenarien können somit anhand der Notizen der testenden Person definiert werden.

3.5. Plausibilitätsprüfung der Testergebnisse

Trotz gewissenhafter Testvorbereitung und -durchführung müssen Rückmeldungen stets auf Plausibilität geprüft und ggf. bestätigt werden. Es gilt auszuschließen, dass es sich um einen Fehler im Testaufbau oder eine False-Positive-Meldung handelt. Vorkommen kann dies insbesondere bei automatisierten Schwachstellen-Scans. Um zu vermeiden, dass Zeit und Energie falsch investiert werden, sollte das Ergebnis auf Plausibilität geprüft werden. Ein Beispiel kann darin bestehen, dass ein Schwachstellen-Scanner eine falsche Versionsnummer erkennt und daraufhin einen Verweis auf eine Schwachstelle erstellt, die ggf. in der real eingesetzten Version bereits behoben wurde.

3.6. Bewertung der Ergebnisse

Mit den belastbaren Testergebnissen einhergehend ist eine Priorisierung vorzunehmen. Anhand dieser können im nächsten Schritt angemessene Maßnahmen zur Behebung der Schwachstellen getroffen werden.

In der hausinternen Kommunikation sollte darauf geachtet werden, dass Ergebnisse von Security-Tests als das wahrgenommen werden, was sie sind, nämlich wertvolle Erkenntnisse zur Qualitätssicherung.

Im Rahmen der Befundbewertung sollte auch eine Einschätzung der Testabdeckung erfolgen. Diese dient als Maß für die Güte der Aussagekraft der durchgeführten Tests.

3.7. Identifikation von Ursachen und Auswahl geeigneter Maßnahmen

Alle gefundenen Schwachstellen sollten priorisiert werden. Entsprechend der Priorisierung wird an der Behebung der Schwachstellen gearbeitet. Im Fokus sollte hier die Beseitigung durch z. B. Behebung der Schwachstelle im Quellcode, Aktualisierung verwendeter Bibliotheken oder die Änderung der Konfiguration stehen. Wenn dies nicht möglich ist, sollten durch Integratoren oder Betreiber organisatorische Maßnahmen ergriffen werden, wie beispielsweise die Beschreibung von Workarounds .

Erfahrungsgemäß sind Maßnahmen unzureichend, um Gefährdungen nachhaltig entgegenzuwirken, wenn sie nur symptomlindernd oder gar verschleiernd wirken.

Nach der Umsetzung der abgeleiteten Security-Maßnahmen sollte ein erneuter Testdurchlauf durchgeführt werden, um sowohl die Wirksamkeit der Maßnahmen zu prüfen als auch negative Auswirkungen auf andere Systemkomponenten ausschließen zu können.

4. Testszenarien

Dieses Kapitel soll bei der Erstellung geeigneter Testszenarien helfen, indem zu ausgewählten Protokollen und Schnittstellen verschiedene Prüffragen aufgeführt werden. Die Prüffragen orientieren sich an den im Kapitel 3 beschriebenen Zielen. Bedingt durch die Komplexität dieses Themas besitzen diese jedoch keinen Anspruch auf Vollständigkeit. Sie sind daher als wertvolle Arbeitshilfe zu verstehen.

Eine Vielzahl der Fragestellungen zielen nicht unbedingt explizit auf die Security ab, helfen aber dabei mögliche Fehler in der Entwicklung oder Konfiguration zu verhindern oder aufzudecken. Einige Fragestellungen sind auch organisatorischer Natur.

4.1. Allgemeine Fragestellungen bei der Umsetzung von Schnittstellen

Alle Schnittstellen, über die Daten angenommen werden, sollten einer Prüfung unterzogen werden. Hierbei gelten, unabhängig von der Technik und dem gewählten Format, einige allgemeine Aspekte, die betrachtet werden sollten:

4.1.1. Auswahlprozess

- Ist die Schnittstelle, das Übertragungsprotokoll oder das Datenformat geeignet, den Schutzbedarf und die Anforderungen der zu verarbeitenden Daten im gegebenen Einsatzgebiet zu gewährleisten?
- Wird jeweils die aktuelle Version des Protokolls genutzt bzw. werden bekannte Schwachstellen berücksichtigt? Wie wird auf neue Schwachstellen reagiert?

4.1.2. Umsetzung der Spezifikationen

- Ist die Implementierung konform zu den Protokoll- / Datenspezifikationen?
- Sind alle verpflichtenden Funktionen der Protokoll- / Datenspezifikationen umgesetzt?
- Sind alle verpflichtenden Konfigurationen der Protokollspezifikationen umgesetzt?
- Bleibt die Implementierung bei der Verarbeitung von Daten für (nicht) umgesetzte Funktionen in einem stabilen Zustand?
- Werden alle Längenbegrenzungen geprüft und korrekt verarbeitet?
- Wird vor der Verarbeitung von Daten geprüft, dass die Daten im richtigen Datentyp vorliegen?

4.1.3. Implementierung

- Werden im Fehlerfall belegte Ressourcen wieder freigegeben?
- Werden Maßnahmen zur Qualitätssicherung des Quellcodes umgesetzt (z.B. durch statische Codeanalyse)?
- Werden nicht benötigte Codebestandteile und Funktionen vor der Auslieferung entfernt?
- Ist ein Berechtigungsmanagement für Nutzer, Daten und Drittkomponenten korrekt implementiert?
- Sind die ausgewählten Softwarekomponenten uneingeschränkt auf der Hardware einsetzbar? Teilweise werden auf unterschiedlichen Hardwareplattformen Funktionen wie Zufallszahlengeneratoren unterschiedlich realisiert und eingebunden.

4.1.4. Schwachstellen-Scanning

- Wird aktiv nach der Veröffentlichung von Schwachstellenmeldungen in genutzten Drittanwendungen/Programmbibliotheken recherchiert?
- Werden diese Erkenntnisse in der eigenen Entwicklung berücksichtigt (z. B. durch Updates der Firmware)?
- Werden neue oder weiterentwickelte Komponenten regelmäßig mittels Schwachstellen-Scanner geprüft?

4.1.5. Fuzzing

- Ist die Protokollimplementierung stabil gegenüber unerwarteten Daten? Dies können z. B. sein,
  - nicht unterstützte Zeichencodierungen,
  - Fehler in serialisierten Datenströmen,
  - spezifikationskonforme Steuerdaten, jedoch ungültige oder unberechtigte Nutzdaten,
  - gültige und berechtigte Nutzdaten ohne spezifikationskonforme Steuerdaten,
  - Daten außerhalb des Wertebereichs,
  - fehlerhaft verschlüsselte / signierte Daten,
  - stimmen Angaben in Längenfeldern mit den Daten überein und wird dies geprüft,
  - veränderte Reihenfolgen von Datenfeldern oder Nachrichten.
- Wurden alle infrage kommenden Übertragungsparameter getestet und verblieben ohne negativen Befund?
- Werden vom Client übermittelte Daten auf Datenlänge und -typ validiert?
- Ist die Implementierung stabil gegenüber Verbindungsabbrüchen in sämtlichen Kommunikationsphasen?

4.1.6. Lasttest / Verfügbarkeit

- Wie verhält sich die Komponente bei Überlast an einer Schnittstelle (z. B. mit zu vielen Anfragen)? Ist dieses Verhalten akzeptabel?
- Wie verhält sich die Komponente bei Latenzen, die bei der Übertragung auftreten?
- Wie verhält sich die Komponente bei gehäuften Verbindungsabbrüchen?
- Wie verhält sich die Komponente bei Ausfall oder Beeinträchtigung von anderen Diensten (z. B. DNS, Routing, usw.)?

4.1.7. Protokollierung

- Lassen sich Änderungen an den Security-Einstellungen in einem Protokoll nachvollziehen und nachweisen?
- Lassen sich Protokolleinträge an einen zentralen Log-Server übertragen?
- Kann die Komponente in ein Monitoring aufgenommen werden?

4.1.8. Konfiguration

- Sind Konfigurationsdaten und deren Backups gegen missbräuchliche Änderungen geschützt?
- Sind ausschließlich die notwendigen und dokumentierten Anwendungen/Dienste/Funktionen verfügbar?
- Wie wird mit Standardzugangsdaten verfahren? Ist beispielsweise ein Wechsel bei der Inbetriebnahme erforderlich?

4.1.9. Schnittstellen

Unter einer Schnittstelle wird hier ein externer Anschluss verstanden, über den mit der Komponente kommuniziert werden kann. Dies können beispielsweise ein Netzwerk- oder USB-Anschluss sein.

- Stehen ausschließlich die erforderlichen physikalischen Schnittstellen zur Verfügung?
- Lassen sich nicht benötigte Schnittstellen nachhaltig deaktivieren? Wird diese Vorgehensweise in der Dokumentation empfohlen und die Vorgehensweise beschrieben?
- Kann das ICS gewährleisten, dass ausschließlich bestimmte Komponenten nach dem Anschluss genutzt werden können?
- Wurden eventuell erforderliche Treiber auf Vertrauenswürdigkeit und Stabilität geprüft?
- Wurden nicht benötigte Treiber deaktiviert?
- Wurden mit Hardwaremanipulationen verbundene Risiken für das ICS betrachtet? Wurden wirksame Maßnahmen ergriffen?
- Wie reagiert die Komponente auf die Unterbrechung einer Verbindung?
  - Werden anfallende Daten zwischengespeichert und später übermittelt?
  - Werden bestehende Verbindungen überwacht um einen Ausfall zu erkennen?
- Sind Maßnahmen zum Schutz der Vertraulichkeit, Integrität und Authentizität der Daten getroffen worden?
- Werden ausschließlich Daten aus berechtigten Quellen verarbeitet?

Vor der Inbetriebnahme in einer Anlage sollte folgendes geprüft werden:

- Wird aufgrund von Fehlkonfiguration die Betriebsaufnahme mit einer aussagekräftigen Meldung verweigert?
- Sind Standardbenutzer und Standardpasswörter in den Voreinstellungen ausgeschlossen oder können diese geändert bzw. die Änderung vor Inbetriebnahme erzwungen werden?
- Wurden Maßnahmen ergriffen, die eine hohe Passwortqualität gewährleisten?
- Sind sonstige Vertrauensstellungen in den Voreinstellungen ausgeschlossen?
- Sind Voreinstellungen auf eine Gewährleistung der Schutzziele ausgerichtet?

4.1.10. Netzwerkdienste

Eine netzwerkbasierte Schnittstelle stellt den am Netzwerk teilnehmenden Systemen einen Dienst über ein Protokoll zur Verfügung. Es werden im folgenden Anregungen für Testszenarien beschrieben:

- Welche Netzwerkdienste warten an den Netzwerkschnittstellen auf eingehende Verbindungen?

- Sind Kommunikationsbeziehungen in der Standardkonfiguration bereits möglichst restriktiv definiert? Sind die Einstellungen weiter änderbar?

4.1.11. Kryptografie

Bei Protokollen, die auf kryptografische Funktionen zurückgreifen, sind unter anderem folgende Aspekte zu berücksichtigen:

- Werden geeignete kryptografische Algorithmen und Parameter (z. B. Schlüssellängen sowie derzeit als sicher geltende) verwendet?
- Wurde bei der Auswahl ein langer Produktlebenszyklus berücksichtigt? D. h. wurde bei z. B. Schlüssellängen die weitere Entwicklung der Rechenleistung berücksichtigt.
- Werden kryptografische Schlüssel sicher gespeichert?
- Besteht die Möglichkeit, dass der Betreiber die Zertifikate und privaten Schlüssel austauscht?

Ein wichtiges Thema ist auch die Zertifikatsvalidierung:

- Werden bei der Verwendung von Zertifikaten Gültigkeitszeitraum, Inhaber (Common Name/Subject Alternative), Aussteller und Status geprüft?
- Werden Zertifikatsparameter, wie der Verwendungszweck für Verschlüsselung, Signatur oder Client- bzw. Serverauthentisierung, geprüft?
- Werden auch die Aussteller-Zertifikate und die Zertifikatskette geprüft?
- Wird der aktuelle Zustand der Zertifikate z. B. mittels Sperrlisten geprüft?
- Wird der Endnutzer bei der Verwendung von HTTPS und nicht in den Browsern hinterlegten Wurzelzertifikaten darauf hingewiesen, auf welche Dinge er zu achten hat (z. B. Vergleich des Fingerprint des Zertifikat im Browser mit der Dokumentation)?
- Können Zertifikate auf eine Whitelist gesetzt werden (z. B. bei Verwendung von selbstsignierten Zertifikaten)?

4.2. Industriespezifische Protokolle

4.2.1. Modbus

Das Modbus-Protokoll ist ein industriespezifisches Protokoll. Es bietet keine Funktionen zur Authentisierung, Verschlüsselung oder Integrität und Authentizität der Daten.

Ergänzend zu den allgemeineren Anregungen aus Kapitel 4.1 werden nachfolgend einige Fragen aufgeworfen, die als Anregungen für protokollspezifische Testszenarien genutzt werden können:

- Wie reagiert die Implementierung auf Befehle, die den Slave in einen passiven Modus versetzen?
- Wie verhält sich die Implementierung bei der Anweisung, die Kommunikation neu zu starten?
- Wie verhält sich die Implementierung bei der Anweisung, die Diagnosezähler zu leeren, zu löschen und zurückzusetzen?
- Verhält sich die Implementierung stabil gegenüber Modbus TCP Paketen mit einer inkorrekt angegebenen Größe oder Länge?
- Wie verhält sich Modbus TCP, wenn andere Daten, Protokolle oder verstümmelte Modbus-Nachrichten übermittelt werden?

- Wie verhält sich die Implementierung, wenn in einer Exception PDU weitere Nachrichten oder Anweisungen enthalten sind?
- Wie stabil bleibt die Implementierung, wenn von einem Server an alle Slaves Nachrichten versandt werden?
- Wie verhält sich die Implementierung, wenn eine Liste der verfügbaren Anweisungen abgerufen wird?

4.2.2. PROFINET

Das „Process Field Network“ (kurz: PROFINET) ist ein modularer Protokollstandard für die Automatisierung.

Die PNO bietet mit [PNO-1] bereits umfangreiche Informationen zum Thema Robustheit. Ergänzend zu den allgemeineren Anregungen aus Kapitel 4.1 werden nachfolgend einige Fragen aufgeworfen, die als Anregungen für protokollspezifische Testszenarien genutzt werden können:

- Weist die PROFINET Implementierung ein normkonformes Verhalten nach IEC 61158 auf?
- Würde die gegenwärtige Implementierung eine Zertifizierung bestehen oder wurde ein Zertifizierungsverfahren bestanden?
- Sind in der Anlage Schutzmaßnahmen gegen DoS-Angriffe und weitere unberechtigte Zugriffe und Fehlzugriffe auf Automatisierungskomponenten etabliert?
- Wie stabil verhält sich die IO-Implementierung gegenüber zufälligen oder manipulierten Daten im PROFINET IO, RPC und UDP Protokoll?
- Wie stabil verhält sich die Implementierung gegenüber zufälligen oder manipulierten Daten im PROFINET DCOM & RPC und TCP Protokoll?
- Welche Auswirkungen hat eine hohe Nutzlast RPC / DCOM Daten auf die angestrebten Echtzeitanforderungen?
- Wird die „General Station Description“ (GSD) Datei im Hinblick auf die Integrität angemessen geschützt?

4.2.3. OPC

4.2.3.1. OPC classic

„OLE Process Control“ (OPC) wird zur Übertragung von Mess- und Steuerdaten genutzt. Es bietet keine Funktionen zur Authentisierung, Verschlüsselung oder Integrität und Authentizität der Daten.

Ergänzend zu den allgemeineren Anregungen aus Kapitel 4.1 werden nachfolgend einige Fragen aufgeworfen, die als Anregungen für protokollspezifische Testszenarien genutzt werden können:

- Konnte in Testszenarien nachgewiesen werden, das auf Schadwirkung ausgerichtete OLE / DCOM Nachrichten erfolgreich erkannt und abgewiesen werden?
- Wurden die DCOM Kommunikationsschnittstellen soweit wie möglich eingeschränkt?
- Bei einheitlicher Verwendung identischer Windows Authentisierungsdaten (Benutzername und Passwort) an beteiligten Systemen: Gestatten die verfügbaren Netzwerkschnittstellen und physikalischen Zugriffsmöglichkeiten weitere Missbrauchsmöglichkeiten? Wurden Maßnahmen zur Reduktion des Risikos getroffen?

4.2.3.2. OPC UA

OPC UA (OPC Unified Architecture, IEC 62541) wird zum Austausch von Daten und Informationen und dem Aufruf von Diensten genutzt. Es bietet Funktionen zur Authentisierung, Verschlüsselung und Integrität bis in die Daten- und Diensteebene.

- Wurden die Securityfunktionen für OPC UA zur Verschlüsselung, Signatur und Authentisierung aktiviert?
- Sind Security-Aspekte zum Einbringen und Wechseln der Zertifikate berücksichtigt?
- Können Zugriffslisten für die Registrierung von Clients auf dem OPC UA Server konfiguriert werden? Sind diese eingerichtet?
- Wurden OPC Client und Server mittels der OPC Compliance Test Tools (CTT) geprüft und ggf. eine Zertifizierung durch die OPC-Foundation vorgenommen?
- Werden Signaturen nicht nur mathematisch auf Korrektheit geprüft, sondern auch bzgl. der Gültigkeit der Zertifikate (z.B. Trusted Issuer)?

4.3. Kommunikationsschnittstellen

4.3.1. USB (Universal Serial Bus)

Universal Serial Bus (USB) ist ein Standard für externe Schnittstellen, der eine Kommunikation zwischen einem System und verschiedenen anderen Peripheriegeräten ermöglicht.

Ergänzend zu den allgemeineren Anregungen werden nachfolgend einige Fragen aufgeführt, die als Anregungen für spezifische Testszenarien genutzt werden können:

- Wird die Nutzung von USB-Geräten eingeschränkt?
- Wurden die Komponenten oder die Gerätetreiber einer Konformitätsprüfung unterzogen und haben sie diese ohne Einschränkungen durchlaufen? Hierfür kann z. B. das vom USB-IF (USB Implementers Forum Inc.) zur Verfügung gestellte USB-Compliance Test-Tool[4] verwendet werden.
- Ist asynchron durchgeführtes Trennen und Hinzufügen der USB-Verbindung zwischen dem USB-Master und dem USB-Slave jederzeit möglich, ohne das die Geräte, der USB-Stack oder die Software in einen undefinierten Zustand geraten?
- Wenn der USB-Master oder der USB-Slave ihren Power-Mode (Laufend → Suspend-Modus und Suspend-Modus → Laufend) während einer aktiven Verbindung ändern, ist dies analog zur Verbindungstrennung ohne Stabilitäts- und Datenverlust möglich?
- Die Verbindung muss gemäß den USB-Standards auf Änderungen der Topologie am USB-Bus vorbereitet sein. Dazu können der USB-Master und der USB-Slave über einen USB-Hub miteinander verbunden werden, an den, analog zur Verbindungstrennung, asynchron andere USB-Geräte angeschlossen und entfernt werden. Auch hier sollte der getestete USB-Stack alle entsprechenden Hotplug-Events verarbeiten und wenn nötig, an die Applikation weiterreichen.

4.3.2. FireWire

FireWire (IEEE1394) wurde für hohe Datenübertragungsraten konzipiert. Es erlaubt, je nach Treiberkonfiguration, auch den direkten Zugriff auf den Speicher des Hosts.

4 http://www.usb.org/developers/docs#comp_test_procedures

- Hat das FireWire-Gerät die FireWire-Konformitätstests komplett und erfolgreich durchlaufen, ohne an Stabilität und Reaktionszeit zu verlieren?
- Wie reagiert die FireWire Implementierung des Testgegenstands und ggf. das FireWire-Gerät auf gefuzzte Daten?
- Ist asynchron durchgeführtes Trennen und Hinzufügen der FireWire-Verbindung zwischen dem Host und dem Slave jederzeit möglich, ohne dass die Geräte, der FireWire-Stack oder die Software in einen undefinierten Zustand versetzt werden?
- Wenn der Host oder der Slave ihren Power-Mode (Laufend → Suspend-Modus und Suspend-Modus → Laufend) während einer aktiven Verbindung ändern, ist dies analog zur Verbindungstrennung ohne Stabilitäts- und Datenverlust möglich?
- Ist ein direkter Speicherzugriff (DMA) über die FireWire-Schnittstelle möglich (lesend / schreibend)? Ein anschauliches Beispiel für einen gezielten Angriff ist der sogenannte „Inception"-Angriff auf FireWire-Schnittstellen[5]. Kann ein Angreifer damit ggf. die Software dauerhaft verändern und so Hintertüren im Gerät hinterlegen?

4.3.3. Wi-Fi

Wi-Fi ist der übergeordnete Begriff für Funkstandards, die auf der IEEE-Norm IEEE-802.11 basieren.

Nachfolgend sind einige Fragen aufgeführt, die als Anregungen für Testszenarien genutzt werden können:

- Wurden Maßnahmen für eine Verschlüsselung der Daten und eine Authentisierung getroffen?
- Wird in der Dokumentation explizit über die ggf. vorhandenen Risiken aufgeklärt? Werden die notwendigen Einstellungen in der Dokumentation erläutert? Sind diese Einstellungen im Auslieferungszustand aktiviert?
- Bei Verwendung von Wi-Fi in ICS mit Betriebssystemen: Speichert das Betriebssystem bisher konfigurierte Netze und Authentisierungsdaten?
  - Ist sichergestellt, dass keine Verbindung zu gefälschten Access-Points hergestellt werden und darüber Daten abgegriffen werden?
  - Werden Zugangsdaten zu konfigurierten (drahtlosen) Netzen im Betriebssystem sicher gespeichert ?

4.3.4. Bluetooth

Bluetooth kommt für drahtlose Verbindungen im Nahbereich zum Einsatz.

- Sind nur die beiden obligatorischen, sowie die für die Kommunikation notwendigen Profile aktiv bzw. verfügbar?
- Ist eine Authentisierung mit zufälligem Schlüssel implementiert?
- Können die Authentisierungsdaten geändert werden bzw. wird nicht für jedes Gerät derselbe Schlüssel verwendet?
- Wie verhält sich der Bluetooth-Stack im Testobjekt bei hohen Datenaufkommen? Verbleibt der Bluetooth-Stack im Testobjekt auch bei massivem Datenverkehr durch Bluetooth Schwachstellen-Scanner stabil?

5 http://www.breaknenter.org/projects/inception/

4.3.5. IEEE 802.15.4 Wireless Personal Area Network

Haupteinsatzgebiet von IEEE 802.15.4 sollen selbstorganisierende Ad-hoc-Netzwerke sein. Dabei sollen sowohl vermaschte als auch Peer-to-Peer-Netzwerke aufgebaut werden können, die selbst wieder zu größeren sogenannten Cluster-Bäumen verbunden werden können. Trotz dieser kommunikationstechnischen Anforderungen an die Geräte soll der Energiebedarf minimal sein. Der Standard wurde in verschiedenen offenen und proprietären Protokollen umgesetzt. Als Beispiele aus dem industriellen Umfeld seien hier ZigBee, WirelessHART oder ISA 100.11a genannt.

- Werden die Funktionen zur Verschlüsselung genutzt?
- Werden die Access Control Lists genutzt?
- Wie reagieren die auf die Übertragungsschicht zugreifenden Anwendungen auf deren Ausfall oder Überlastung?
- Bleiben die Kommunikationspartner jeweils stabil, wenn mittels Fuzzing die Datenkommunikation unerwartete Inhalte aufweist?
- Kann in der Implementierung des Testobjekts eine Nachricht mit einem sehr hohen Frame Counter durch einen Kommunikationspartner oder Dritte erfolgreich übermittelt werden? Welche Auswirkungen sind damit verbunden?

4.4. Anwendungsschnittstellen

4.4.1. HTTP/HTTPS

Das Hypertext Transfer Protocol (HTTP) bzw. die sichere (zu bevorzugende Variante) HTTPS (vgl. Maßnahmen M 36) mit der Transportverschlüsselung und Authentisierung des Servers wird hauptsächlich zur Übertragung von Webseiten oder in Webservices (bei SOAP oder REST) eingesetzt.

- Werden von den möglichen HTTP Request Methoden GET, POST, HEAD, PUT, DELETE, OPTIONS, TRACE und CONNECT ausschließlich die erforderlichen Methoden unterstützt?
- Da HTTP TRACE für Cross-Site-Scripting Attacken missbraucht werden kann: wird die Unterstützung dieser HTTP Methode wirksam verhindert?
- Wurden potentielle Gefährdungen bei der HTTP-Methode CONNECT, wie Aneinanderreihung von HTTP Proxys, ausreichend betrachtet?
- Werden serverseitig Felder im HTTP Response stets korrekt generiert (insbesondere Content-Type, Content-Lenght, Content-Encoding und Transfer-Encoding)?
- Werden HTTP Requests auf relative Pfadangaben korrekt abgefangen (z. B. HTTP GET /subdir/../../)?

Anregungen für webbasierte Anwendungen:

- Wurden Maßnahmen in sämtlichen Prozessen ergriffen, um die zustandslose Eigenschaft des HTTP Protokolls zu kompensieren?
- Werden die Authentisierung und Autorisierung von Zugriffen auf Dateien, Verzeichnissen, Daten und web-basierten Funktionen stets geprüft? Auch wenn die angeforderte Ressource vermeintlich nicht anderweitig verlinkt ist?
- Liegen neben der web-basierten Anwendung keine weiteren temporären Daten, Datensicherungen, Standarddokumente und -scripte des Webservers in den Verzeichnissen?
- Ist die Webanwendung gegen Referer Spoofing resistent, sodass der von Clients im HTTP Request angegebene Referer zu keiner Zeit den logischen Ablauf beeinflusst?

- Werden Session IDs nach einem tatsächlich zufälligen Muster generiert?
- Werden Session IDs im vorgesehenen Anwendungsablauf niemals über die URL geführt?
- Wird in webbasierenden Anwendungen niemals ein Client automatisch als authentisiert angenommen, selbst wenn scheinbar im HTTP Request bestimmte Eigenschaften erfüllt wirken (z. B. Hostname = superuser.kunde.de, Client = UnserAdministrationClient v2)?
- Ist die webbasierte Anwendung resistent gegenüber Cookie Manipulationen?
- Ist die webbasierte Anwendung bei deaktivierten Cookies weiterhin nutzbar?
- Berücksichtigt die Logik der webbasierten Anwendung, dass die Vorgabe zur Lebensdauer eines Cookies von der Gegenseite nicht unbedingt eingehalten wird?

4.4.2. FTP

Das „File Transfer Protocol“ (kurz: FTP) ist ein Protokoll zur Übertragung von Dateien zwischen einem Client und Server.

Mit „FTP over SSL“ (kurz: FTPS) wurde eine zusätzliche Protokollschicht zwischen TCP und FTP eingeführt, um Authentisierung und Verschlüsselung zwischen FTP-Client und FTP-Server zu erreichen.

Das erste Ziel wird durch einen TLS-Handshake mit Validierung des Zertifikats am Client und/oder Server erreicht. Das zweite Ziel wird anschließend über kryptografische Verschlüsselung mit einem als sicher anerkannten Verschlüsselungsstandard unter Verwendung der validierten Zertifikate erfüllt.

Bei dem „SSH File Transfer Protocol“ (kurz: SFTP oder auch als Secure FTP bekannt) handelt es sich um eine FTP-Implementierung, die auf SSH für die Verwaltung und den Transfer von Daten setzt. SFTP lässt sich leichter in Security-Infrastrukturen einführen. Das Protokoll benötigt – im Gegensatz zu FTPS – eine einzige Verbindung zwischen Client und Server.

4.4.2.1. Anregungen für serverseitige Testszenarien

- Wurde die Wahl zwischen aktivem und passivem FTP auf den bedarfsgerechten Betrieb von Security-Infrastrukturen beim Endnutzer abgewogen?
- Ist eine dem Sicherheitsniveau angepasste Authentisierung implementiert worden?
- Wurden Maßnahmen ergriffen, die gegen Brute-Force-Angriffe auf die Authentisierungsdaten erschwerend wirken und zugleich die betroffene Rolle / den FTP-Benutzer nicht vollständig für inakzeptable Dauer sperren?
- Sind Lese- und Schreibzugriffe auf Verzeichnisse und Dateien nachhaltig an Benutzer / Rollen gebunden und lassen sich nicht umgehen?
- Wurde auf die Notwendigkeit verschlüsselter Kommunikation zwischen Client und Server geprüft?
- Lässt sich der FTP-Server ausschließlich durch berechtigte Clients erreichen?
- Ist der FTP-Server frei von bekannten Schwachstellen?
- Wurde der FTP-Server darauf ausgerichtet, möglichst wenig Informationen über seine Bezeichnung und Versionsnummer auszugeben?
- Ist der Server resistent gegenüber Directory Traversal Angriffen (z. B. cd subdir/../../../)?
- Ist der FTP-Server selbst unter Lastspitzen, unerwarteten Verbindungen und Befehlen stabil?
- Unterstützt der FTP-Server ausschließlich die zur Diensterfüllung erforderlichen Befehle?

- Wird verhindert, dass dieselben Accountdaten für den sicheren (SFTP, FTPS) und unsicheren (FTP) Einsatz genutzt werden?

4.4.2.2. Anregungen für clientseitige Testszenarien

- Kann der Client mit Verbindungsabbrüchen stabil umgehen?
- Werden dem Schutzbedarf entsprechend wirksame Maßnahmen umgesetzt, um lokal gespeicherte Daten zu schützen?
- Bleibt der Client bei unerwarteten Daten-Encoding, Antworten und Antwortzeiten stabil?

5. Mapping zu ISO/IEC 62443-4-1 und ISO/IEC 62443-4-2

Das dargestellte Mapping der Maßnahmen erfolgte anhand des Entwurfs der ISO/IEC 62443-4-1 mit Stand Juni 2014. Eine Abbildung auf die Norm ISO/IEC 62443-4-2 erfolgt sobald das Dokument eine stabile Form erreicht hat.

ICS-Komplendium	ISO/IEC 62443-4-1
M1	SMP-1.1, MIV-1, SIT-1
M2	SMP-1.4
M3	MIV-3
M4	SIT-2
M5	SRS-2
M6	
M7	
M8	DSG-1.6, SRP-1
M9	MIV-9, SRP-2.1
M10	SMP-1.1
M11	Sre-1,SRE-3
M12	
M13	
M14	
M15	SMP-1.1, SRS-1, SAD-5
M16	SRS-3, SAD-2, DSG-1
M17	DSD-2
M18	DSD-2
M19	
M20	
M21	
M22	
M23	
M24	MIV-7
M25	MIV-7
M26	DSD-3
M27	
M28	
M29	
M30	
M31	
M32	
M33	

M34	
M35	
M36	
M37	
M38	
M39	
M40	MIV-7
M41	(SAD-1)
M42	
M43	

6. Übersicht zu nutzbaren Werkzeugen

Für den Einstieg in das Thema stellt die Linux-Distribution „Kali“ eine Vielzahl an Werkzeugen bereit. Die nachfolgende Übersicht stellt einige dieser Werkzeuge vor. Neben den Open-Source-Anwendungen gibt es eine Vielzahl kommerzieller Produkte vergleichbaren Umfangs.

Die Werkzeuge unterstützen im wesentlichen nur in der Office-IT gebräuchliche Protokolle. Für spezielle Industrieanwendungen fehlen bisher entsprechende Werkzeuge.

6.1. Schwachstellen-Scanner

Schwachstellen-Scanner bieten die Möglichkeit, Systeme und Anwendungen automatisiert zu prüfen. Sie besitzen Datenbanken, in denen bekannte Schwachstellen enthalten sind oder Funktionen um diese zu erkennen.
Der Umfang der Tests variiert. Es gibt Scanner, die unterschiedliche Anwendungen vom Betriebssystem bis zu einzelnen Anwendungen prüfen können. Diese werden im ersten Unterkapitel behandelt. Danach folgen solche, die auf die Webschnittstelle angepasst wurden.

6.1.1. Allgemein

6.1.1.1. OpenVAS

„Open Vulnerability Assessment System“ (kurz: OpenVAS) ist ein leistungsfähiger Schwachstellen-Scanner und Schwachstellenmanager, der bereits bekannte Schwachstellen finden kann. Dies kann z. B. in verwendeter 3rd-Party-Software der Fall sein. Bei Eigenentwicklungen ist er deutlich weniger leistungsfähig, da hierfür keine Informationen vorliegen.

Homepage: http://www.openvas.org

6.1.2. Web-Anwendungen

6.1.2.1. OWASP Zed Attack Proxy

Mit dem OWASP Zed Attack Proxy steht ein Werkzeug für HTTP(S)-basierende Anwendungen zur Verfügung. Kernfunktion ist dabei ein HTTP(S)-Proxy zum Mitschneiden und Manipulieren der Kommunikation. Weiter enthalten ist ein Schwachstellen-Scanner und ein HTTP(S) Request-Fuzzer zur Prüfung der Eingabevalidierung und Verarbeitung.

Homepage: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

6.1.2.2. w3af

w3af ist fokussiert auf die automatisierte Identifikation von Schwachstellen von Webanwendungen. Das Werkzeug bringt dazu bereits eine Auswahl an vorgefertigten Profilen mit, die den Einstieg vereinfachen und ohne weitere Konfiguration genutzt werden können.

Homepage: http://w3af.org

6.1.2.3. sqlmap

Für Anwendungen, die im Hintergrund eine Datenbank nutzen, ist das Werkzeug "sqlmap" interessant. Hiermit besteht die Möglichkeit automatisiert auf die Anfälligkeit von SQL-Injections zu prüfen.

Es bietet sich die Nutzung in Verbindung mit z. B. OWASP Zed Attack Proxy an. Dieser kann mittels seines Webcrawlers URLs automatisch sammeln und an sqlmap übergeben .

6.1.2.4. XSSer

Cross-Site Scripting stellt eine der häufigsten Bedrohungen für Webanwendungen dar. Dies tritt dann auf, wenn die Anwendung Daten der Benutzer annimmt und diese dem Browser ohne Validierung und Kodierung weitergibt. Die meisten XSS-Schwachstellen können mithilfe von automatisierten Tests und Code-Analyse gefunden werden.

Ein Werkzeug, das eine Anwendung auf XSS-Schwachstellen automatisiert testet, ist das aus dem OWASP XSSer Project entstandene Cross Site Scripting Framework (kurz XSSer).

Homepage: http://xsser.sourceforge.net/

6.2. Fuzzer

Unter dem Begriff Fuzzing versteht man das Testen des Verhaltens einer Anwendung auf zufällige Eingaben. Dies können Dateien oder auch Netzwerkdaten sein. Das Spektrum reicht dabei von vollständig zufälligen Daten bis hin zu spezialisierten Fuzzern für einzelne Protokolle, in denen nur einzelne Felder geprüft und auch die Erstellung von Prüfsummen berücksichtigt werden.

6.2.1. Allgemein

6.2.2. Peach Fuzzer

Peach ist ein leistungsfähiger Fuzzer zum Generieren und Versenden sowohl von statusfreien als auch von statusbasierenden Nachrichten. Besonders hervorzuheben ist, dass Datenspezifikationen im Peach Fuzzer mittels XML definiert werden und Programmierkenntnisse in C# zwar hilfreich jedoch nicht zwingend erforderlich sind.

Der Peach Fuzzer kann:

- zufällige Daten anhand einer vorgegebenen und einzuhaltenden Protokollspezifikation erstellen und versenden
- zufällige Daten anhand einer vorgegebenen Protokollspezifikation mit validen Feldwerten erstellen und in zufälliger Reihenfolge versenden
- zufällige Daten auf Basis korrekter Datenpakete mit ansteigender Mutation erstellen und versenden

Desweiteren bietet Peach (anders als andere Fuzzer) viele Möglichkeiten der Überwachung, Einschätzung und Reproduzierbarkeit.

Das Tool kann verschiedene Dateiformate und COM/DCOM fuzzen.

Homepage: http://peachfuzzer.com

6.2.3. IP

6.2.3.1. fuzz_ip6

„fuzz_ip6“ ist ein simpler und leistungsfähiger IPv6 Protokoll-Fuzzer ohne graphische Oberfläche. Er kann dazu eingesetzt werden, den Protokollstack auf seine korrekte und fehlertolerante Funktionsweise zu testen.

Homepage: https://www.thc.org/thc-ipv6/

6.2.4. Bluetooth

6.2.4.1. pwntooth

Das Toolset pwntooth wurde entwickelt, um unter Linux Tests gegenüber Bluetooth Geräten automatisiert durchführen zu können. Alle durch das Toolset unterstützten Tools werden mit pwntooth mitgeliefert.

6.3. Kryptografie

6.3.1. SSL /TLS

6.3.1.1. sslyze

Für die Prüfung von Schnittstellen, die TLS nutzen, bietet sich sslyze an. Das Tool überprüft, welche Optionen für die TLS-Verbindungen zur Verfügung stehen und ermöglicht dabei anhand der Ergebnisse etwaigeFehlkonfigurationen zu erkennen und zu korrigieren.

6.4. Statische Codeanalyse

Eine Übersicht von Werkzeugen zur statischen Codeanalyse ist unter [OWASP-1] zu finden. Aufgrund der unterschiedlichen Programmiersprachen und der Entwicklungsumgebungen wird hier keine Empfehlung zu Werkzeugen gegeben.

Abkürzungsverzeichnis

BSI	Bundesamt für Sicherheit in der Informationstechnik
CERT	Computer Emergency Response Team
CSRF	Cross Site Request Forgery
CVE	Common Vulnerability Enumeration
DCOM	Distributed Component Object Model
FTP	File Transport Protocol
HMI	Human Machine Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
ICMP	Internet Control Message Protocol
ICS	Industrial Control System
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IPSec	Internet Protocol Security
ISO	International Organization for Standardization
JTAG	Joint Test Action Group
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OPC UA	OPC Unified Architecture
PNO	Profibus Nutzerorganisation e.V.
RPC	Remote Procedure Call
SNMP	Simple Network Protocol
SOAP	Simple Object Access Protocol
SPI	Serial Peripheral Interface
SPS	Speicherprogramierbare Steuerung
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPM	Trusted Plattform Module
UDP	User Datagram Protocol
USB	Universal Service Bus
UTF-8	8-Bit UCS Transformation Format
VDE	Verband der Elektrotechnik, Elektronik und Informationstechnik
VDI	Verein Deutscher Ingenieure
VPN	Virtual Private Network
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

Literaturverzeichnis

62443-4-1	ISO/IEC: 62443-4-1 Product development requirements
ACATECH-1	acatech: agendaCPS - Integrierte Forschungsagenda Cyber-Physical Systems,
BSI 100-3	BSI: BSI-Standard 100-3: Risikoanalyse auf der Basis von IT-Grundschutz
BSI-100-2	BSI: BSI-Standard 100-2: IT-Grunschutzvorgehensweise, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/standard_1002_pdf.pdf?__blob=publicationFile
BSI-BioKeyS	BSI: BioKEyS - Kryptografisch-biometrische Authentisierungssysteme mittels biometrischer Template-Protection-Verfahren, https://www.bsi.bund.de/DE/Themen/Biometrie/BSIProjekte/BioKeyS/BioKeyS_node.html
BSI-FLOSS	BSI: Freie Software (FLOSS: Freier, Libre und Open Source Software), https://www.bsi.bund.de/DE/Themen/weitereThemen/FreieSoftware/index_htm.html
BSI-Glossar	BSI: Glossar Cybersicherheit, https://www.bsi.bund.de/DE/Themen/Cyber-Sicherheit/Glossar/cs_Glossar_node.html
BSI-GS	BSI: IT-Grundschutz-Kataloge, https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/itgrundschutzkataloge_node.html
BSI-ICS	BSI: ICS-Security-Kompendium, https://www.bsi.bund.de/ICS-Security-Kompendium
BSI-TR-02102	BSI: Kryptographische Verfahren: Empfehlungen und Schlüssellängen, https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html
BSI-TR2102	BSI: Kryptographische Verfahren: Empfehlungen und Schlüssellängen, https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr02102/index_htm.html
BSI-WEB	BSI: Leitfäden zur Entwicklung sicherer Webanwendungen, https://www.bsi.bund.de/DE/Publikationen/Studien/Webanwendungen/index_htm.html
CS-E-Schwachstellen	BSI: Handhabung von Schwachstellen, https://www.bsi.bund.de/ACS/DE/_downloads/techniker/programmierung/BSI-CS_019.pdf?__blob=publicationFile
CS-E-TOP10	BSI: ICS TOP 10 Bedrohungen und Gegenmaßnahmen, https://www.bsi.bund.de/ACS/DE/downloads/techniker/hardware/BSI-CS005.pdf?blob=publicationFile
CWE-TOP25	Common Weakness Enumeration: Top 25 Most Dangerous Software Errors, http://cwe.mitre.org/top25/
GS-OSS	Nikolaus Lefin: IT-Grundschutz-Profil für Open-Source-Software, http://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Hilfsmittel/Extern/Diplomarbeiten/Erstellung_IT-Profil_Lefin.pdf?__blob=publicationFile
ISA-EDSA	ISA Secure: Embedded Device Security Assurance, http://isasecure.org/ISASecure-Program/EDSAAA-Certification.aspx
ISOIEC-62443-2-1	ISO/IEC: 62443-2-1 Reqirements fpr an IACS security management system Ed. 2.0 Profile of ISO27001/27002
OSSTMM	ISECOM: Open Source Testing Methodology Manual, http://www.isecom/research/osstmm.html
OWASP-1	OWASP: Static Code Analysis, https://www.owasp.org/index.php/Static_Code_Analysis
OWASP-2	OWASP: OWASP Testing Guide, https://www.owasp.org/index.php/OWASP_Testing_Project
PNO-1	PROFIBUS Nutzerorganisation e.V.: Test Specification PROFINET IO Security Level 1 / Netload
SANS-Top25	SANS: Top 25 Most Dangerous Software Errors, http://www.sans.org/top25-sofwtare-errors/
VDI2182-1	VDI/VDE: VDI/VDE 2182 Informationssicherheit in der industriellen Automatisierung Blatt 1,