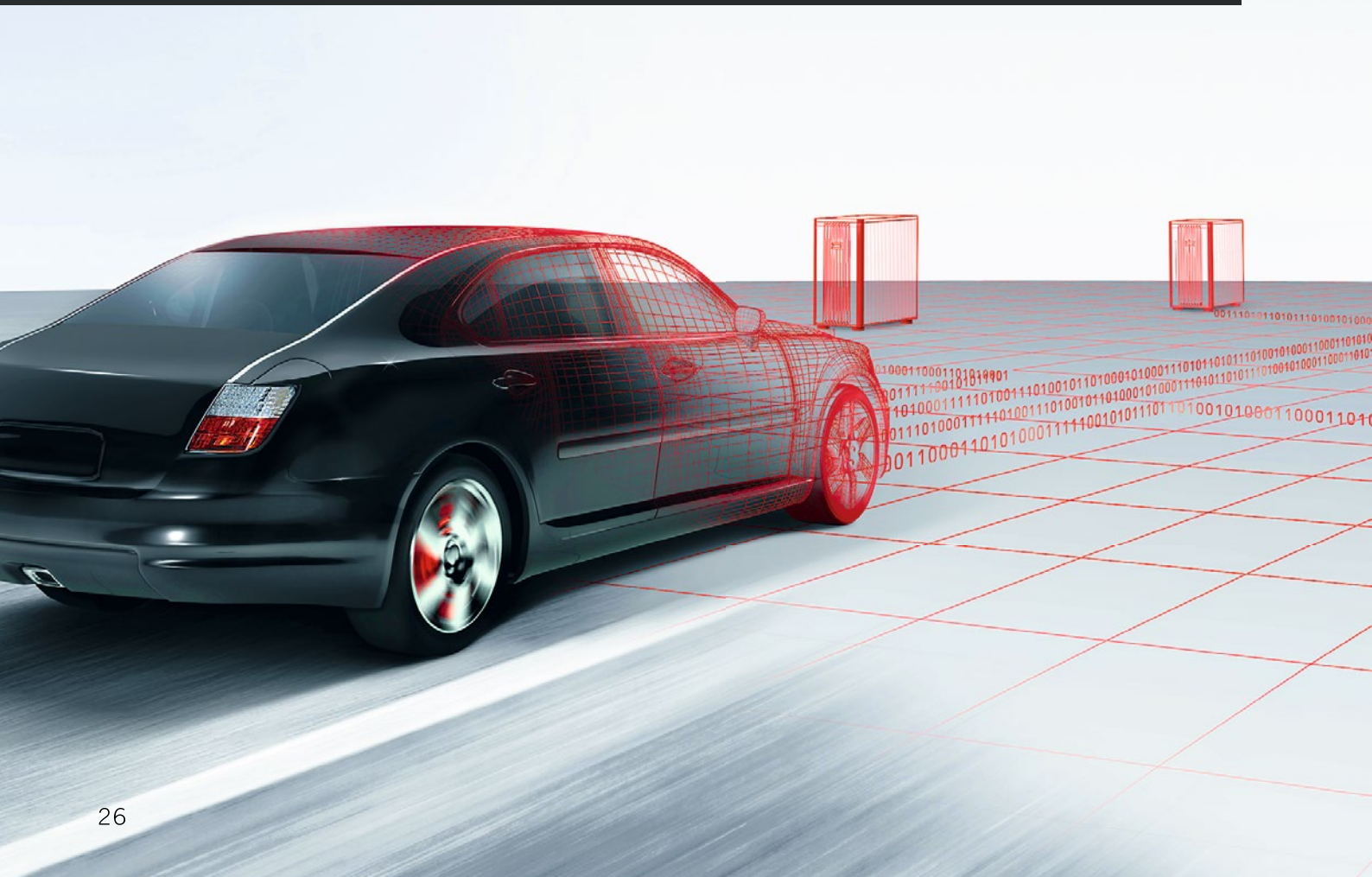
# Methoden zur Umsetzung von **Datensicherheit und Datenschutz im vernetzten Steuergerät**

Der Software-Anteil im Fahrzeug steigt rasant und damit die Anzahl der IT-basierten Schnittstellen wie WLAN, Bluetooth und Mobilfunkzugänge, die das moderne Fahrzeug potenziell angreifbar machen. Die Branche beschäftigt sich deswegen immer stärker mit Konzepten und Lösungen, die das „mobile IT-System“ Pkw schützen. Viele der Lösungsansätze basieren auf modernen mathematischen Ansätzen der Kryptografie, die dazu beitragen, die Vertraulichkeit und Echtheit beziehungsweise Unverfälschtheit von elektronischen Daten nachzuweisen. Aber kann eine einzelne Kryptofunktion oder ein einzelnes Sicherheitsmodul ein Fahrzeug überhaupt angemessen schützen? Dieser Frage geht die Secunet Security Networks AG nach.



AUTOR

**Harry Knechtel**
ist Bereichsleiter Automotive Security bei der Secunet Security Networks AG in München.

## AUFGABENSTELLUNG

Eine alter Lehrsatz des Projektmanagements heißt: Je später ein Problem erkannt wird, desto höher sind die Kosten, um es zu beheben. Dies gilt insbesondere für das Thema IT-Security. Andererseits ist es so, dass Angriffe auf die IT-Sicherheit nur selten dadurch erreicht werden, dass ein einzelner Algorithmus oder Schlüssel gebrochen wird. Aber gerade vor dem Hintergrund personalisierter Dienste, der Integration mobiler Endgeräte und fortgeschrittener Active-Safety-Systeme werden die Themen Datenschutz und Datensicherheit auch für OEMs und Zulieferer immer relevanter. Damit verbunden steigt der Bedarf, die zugrundeliegenden Systeme zu schützen.

Die Aufgabenstellung, eine sichere Software(SW)-Entwicklung zu betreiben, die resistent gegen Angriffe von außen ist, ist nicht trivial. Vor diesem Problem stehen selbst Großkonzerne wie Microsoft, Apple und Co., die sich auf Software-Entwicklung spezialisiert haben. Und auch die Open-Source-Gemeinde, die sich primär um Security-Algorithmen kümmert, ist nicht vor Fehlern gewappnet, wie die erfolgreichen Angriffe und „Exploits“ der näheren Vergangenheit zeigen (Apple SSL Bug, Heartbleed etc.).

Die Aufgabenstellung, ein sicheres System zu entwerfen, zu entwickeln und zu pflegen, bedarf daher eines methodischen Ansatzes. Für die funktionale Entwicklung ist dies bei den OEMs vielfach bereits geregelt – zum Beispiel anhand des V-Modells [1]. Für die weitere Betrachtung lässt sich das V-Modell in die folgenden hier relevanten Phasen einteilen, **BILD 1**:

- Konzeptphase (3)
- Produktentwicklung (4) bis (6)
- Produktion und Betrieb (7).

Im Folgenden wird näher betrachtet, wie sich die Methoden, die Datenschutz und Datensicherheit gewährleisten, in dieses Modell integrieren lassen. Dabei liegt der Fokus auf diesen drei Bereichen:

- Anforderungsanalyse
- sicherer Software-Entwicklungslebenszyklus
- Penetrationstests.

## ANFORDERUNGSANALYSE

Aus erwähnten Gründen muss auch das Thema IT-Security bereits von Beginn an im Entwicklungsprozess mit berücksichtigt werden. In diesem Kontext sind die Themen Datenschutz und Datensicherheit durchaus getrennt zu betrachten.

## ANFORDERUNGSANALYSE – DATENSCHUTZ

Im Sinne des Datenschutzes sollte die Fragestellung zunächst sein: Sind in meinem System personenbezogene und/oder personenbeziehbare Daten vorhanden? Personenbezogen bezieht sich dabei auf Daten mit direktem Bezug zu einer natürlichen Person, beispielsweise Name, Adresse, Alter, Geschlecht. Dahingegen sind personenbeziehbare Daten solche, die erst nach einer Auswertung/Analyse oder einer Kombination/Kumulation Rückschlüsse auf einee Person zulassen. Das können Login- oder Account-Daten, aber auch Navigationsziele, Geo-Positionen oder Anruflisten sein.

Gerade in Bezug auf Datenschutz besteht hier eine Anforderung an Datensparsamkeit. In Deutschland sind Datensparsamkeit und Datenvermeidung in § 3a Bundesdatenschutzgesetz (BDSG) festgeschrieben. Demnach muss genau geprüft werden, warum die Daten erfasst, gespeichert und verarbeitet werden und zu welchem Zweck dies geschieht (Zweckbindung). Dies steht zum Teil im Gegensatz zu einem funktionalen Ansatz, bei dem zunächst genügend Daten erfasst werden und anschließend entschieden wird, welche Funktionen diese Daten erfüllen sollen.

Im nächsten Schritt ist zu definieren, wer, wann und zu welchem Zweck Zugriff auf diese Daten erhalten darf. Der Kunde selbst? Der OEM? Der Zulieferer? Die Vertragswerkstätte oder Drittanbieter? Ist dies festgelegt, geht es um die notwendigen Maßnahmen zum Schutz der personenbezogenen Daten gemäß den definierten Anforderungen und damit in die Betrachtung der Datensicherheit.

## ANFORDERUNGSANALYSE – DATENSICHERHEIT

Vom Thema Datensicherheit sind nicht nur die Daten im Sinne des Datenschut-

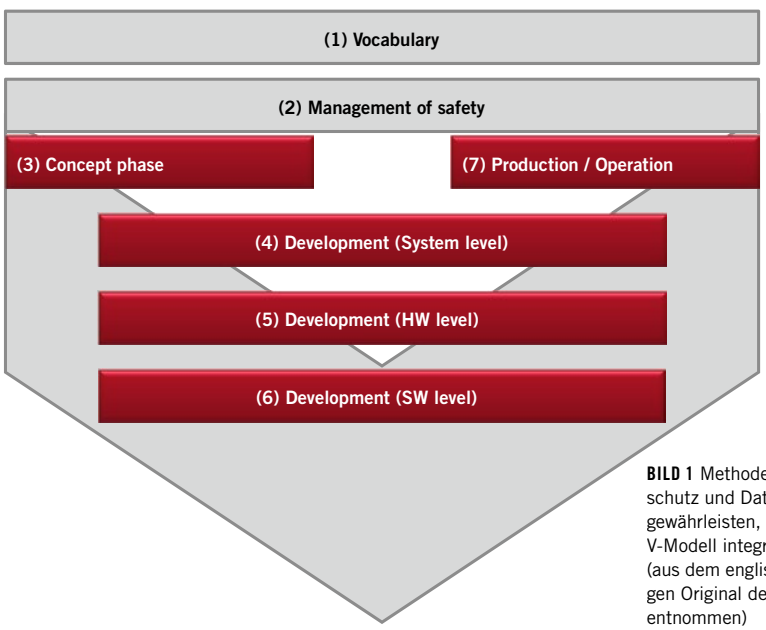



**BILD 1** Methoden, die Datenschutz und Datensicherheit gewährleisten, müssen im V-Modell integriert werden (aus dem englisch sprachigen Original der ISO 26262 entnommen)

zes, sondern alle für das Steuergerät relevanten Daten betroffen. Darin inkludiert sind also auch das Betriebssystem, der Applikations-Code, die Codier- und Parametierdaten oder auch gegebenenfalls verwendete kryptografische Schlüssel und Verfahren.

Für jedes Element muss geprüft werden, wie kritisch diese Daten bezüglich eines möglichen Schadens oder auch der Relevanz für die Safety sind. Daraus werden anschließend die einzelnen Schutzziele hinsichtlich Verfügbarkeit, Integrität, Vertraulichkeit und Authentizität (VIVA-Kriterien) definiert.

Verfügbarkeit bedeutet dabei, dass Daten vorhanden sein müssen, um eine Funktion auszuführen (immer, zeitweise, nie). Integrität ist die Forderung, dass Daten unverändert und korrekt im System vorhanden sind. Vertraulichkeit ist der Schutz von Daten gegen Kenntnisnahme durch unberechtigte Dritte. Authentizität ist die Anforderung, dass Daten wirklich von der angenommenen Quelle stammen.

Um daraus hinreichende und angemessene technische Maßnahmen (Einsatz kryptografischer Algorithmen, Nutzung von Hardware-Sicherheitsmodulen, Separierung/Virtualisierung o.Ä.) ableiten zu können, muss letztendlich ein Angreifermodell definiert werden. Geklärt werden muss vor allem die Frage: Gegen welche Angreifergruppe soll geschützt werden?

Mögliche Fragen sind hierbei beispielsweise: Ist intendiert, den Fahrer des Fahrzeugs mit wenig Know-how und geringen monetären Mitteln von kritischen Funktionen fernzuhalten? Oder ist die Abgrenzung gegen die ungleich größere Anzahl von Hackern und Skript-Kiddies das Ziel, wenn diese zwar über ebenfalls wenig Budget verfügen, aber viel Zeit haben und in großer Masse auftreten? Oder sollen Daten und Systeme gar gegen andere Konzerne mit großen finanziellen Mitteln und Know-how geschützt werden?

Erst aus der Betrachtung dieser verschiedenen Aspekte und unter Einbeziehung der eigentlichen funktionalen Anforderungen an das System ist ein OEM in der Lage, die notwendigen Schutzmaßnahmen an eine Ziel-Lösung zu definieren. Im V-Modell, **BILD 1**, ist dies immer noch die Konzeptphase des Systems.

## SW-ENTWICKLUNGS-LEBENSZYKLUS

Sichere Software-Entwicklung bedeutet in der Praxis mehr als eine Entwicklerrichtlinie. Sichere Software-Entwicklung ist mit organisatorischen, technischen und administrativen Maßnahmen verbunden. Dazu zählt neben grundsätzlichen Security-Trainings für die Entwicklungsingenieure beispielsweise auch die Schaffung eigener Security Crypto Advisors, die sich speziell um das Thema IT-Sicherheit in der Entwicklung kümmern. Diese Advisors können als Kontrollinstanz, aber auch als Ratgeber fungieren.

In der klassischen Software-Entwicklung existieren verschiedene Modelle, die als Grundlage für eine sichere Software-Entwicklung im Embedded-Bereich herangezogen werden können.

Standardisierte Methoden für sichere Software sind zum Beispiel:

- CLASP – Comprehensive, Lightweight Application Security Process
- MS SDL – Microsoft Security Development Lifecycle
- Touch point – Gary McGraw's und das Cigital's Model.

Es gibt aber auch einige nicht kommerzielle Treiber für sichere Software, unter anderem:

- ISSECO – Standardising Secure Software Engineering
- OWASP – Open Web Application Security Project
- SAFECode – Software Assurance Forum for Excellence in Code
- (ISC)² Foundation.

Im V-Modell, **BILD 1**, findet sich dies vor allem in den Phasen (4), (5) und (6) der

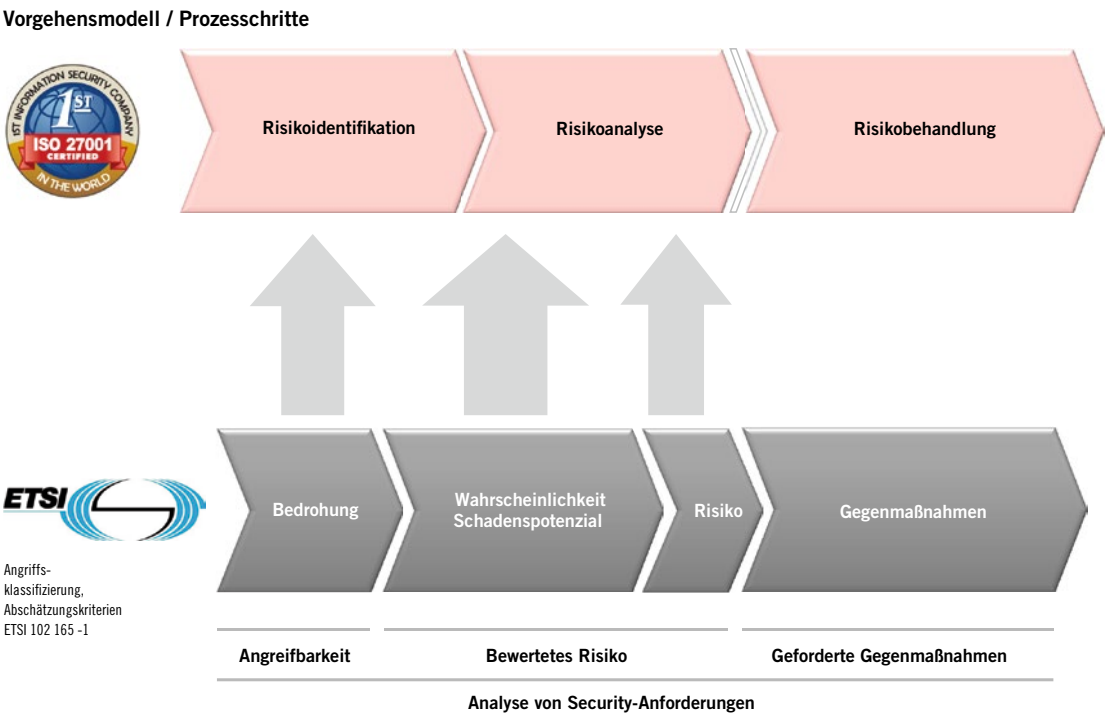



**BILD 2** Neben dem funktionalen Testen ist für sichere Software-Entwicklung ein spezifisches Risk-based Security Testing zu etablieren

Produktentwicklung. Darüber hinaus werden aber auch Teilbereiche aus der Konzeptphase (3) und aus Phase (7) Produktion und Betrieb mit adressiert. Allgemein lässt sich sagen, dass sich die Maßnahmen zur Einführung eines sicheren Software-Entwicklungslebenszyklus immer auf die im Folgenden aufgeführten Bereiche beziehen.

### ANFORDERUNGSANALYSE

Die Anforderungsanalyse in dem Softwareentwicklungszyklus (vergleiche vorheriger Abschnitt) sollte immer im Rahmen einer Zusammenarbeit zwischen den eigentlichen Funktionsverantwortlichen und dem Security Advisor beziehungsweise einem spezifischen IT-Security geschulten Verantwortlichen stattfinden. Gegebenenfalls kann es auch notwendig sein, den Ansprechpartner aus dem Bereich Datenschutz hinzuzuziehen – sofern personenbezogene Daten betroffen sind. Im V-Modell ist dies noch die Konzeptphase (3).

### DESIGN

Im Softwareentwicklungszyklus sollte die Software-Architektur im gesamten Projektteam und unter Einbeziehung des Security Advisors diskutiert werden. Eventuell ist auch die Einführung eines eigenen Bereichs zum Review und zur Freigabe von Zielarchitekturen sinnvoll. Oder es werden Muster-Templates für Standardarchitekturen vorgegeben. Häufig existieren solche Vorgaben oder Prozesse schon aus funktionaler Sicht – oft gibt es aber Verbesserungspotenzial in Richtung der IT-Security-Anforderungen.

Zudem sollte bereits im Design ein Bedrohungsmodell aufgestellt werden (zum Beispiel ETSI TS 102 165-1). In diesem Modell wird beschrieben, wie die angestrebten Schutzziele aus der Anforderungsanalyse durch technische Maßnahmen aus dem Design adressiert werden und welche Restrisiken verbleiben.

### IMPLEMENTIERUNG

Für die Implementierung empfiehlt sich zunächst die Vorgabe von Secure-Coding-Richtlinien. Zum Teil existieren hier schon Vorgaben aus dem Bereich der funktionalen Sicherheit (beispielsweise abgeleitet aus IEC 61508, ISO 26262, IEC 61511, ISO 13849 etc.). Diese sind allerdings durch Vorgaben, welche die spezifischen Anforderungen der IT-Security berücksichtigen, zu ergänzen. Zusätzlich können automatisierte Tools zur statischen Source-Code-Analyse verwendet werden, um „Flüchtigkeitsfehler“ zu entdecken. Besser ist es jedoch, explizite Code Reviews bei sensitiven Funktionen durchzuführen.

### TEST

Neben dem funktionalen Testen ist für sichere Software-Entwicklung ein spezifisches Risk-based Security Testing zu etablieren. Dabei geht es um die Prüfung der spezifizierten Sicherheitsfunktionalität und Abwehrmechanismen durch Insider mit internen Kenntnissen der Systemarchitektur.

Es ist zu beachten, dass es sich hier im Wesentlichen um Negativ-Tests handelt, die in der Regel weniger deterministisch sind als klassische Positiv-Tests einer Funktion. Das bedeutet, dass keine vollständige Testabdeckung möglich ist, da nicht alle möglichen und denkbaren Fehlerfälle getestet werden können. Darum muss eine sinnvolle und angemessene Auswahl an relevanten Security-Tests getroffen werden.

Eine weitere Problemstellung kann zudem darin bestehen, dass Sicherheitsfunktionen zum Teil erst sehr spät in der eigentlichen Entwicklung aktiviert werden (das heißt, Sicherheitsfunktionen sind abgeschaltet oder arbeiten mit Default-Datensätzen), um bis dahin die funktionale Implementierung nicht zu behindern. Auch dieser Umstand muss bei der Testplanung entsprechend berücksichtigt werden.

Neben der Abdeckung während der Entwicklungsphase durch die zuvor skizzierten Risk-based-Security-Tests, **BILD 2**, und das eigentliche funktionale Testing, empfiehlt sich gerade im fortgeschrittenen Entwicklungsstadium für komplexe Steuergeräte der sogenannte Penetrationstest (kurz Pen-Test).

Dabei wird die Software in der Regel wie eine Blackbox auf Sicherheitsschwachstellen durch Angreifer mit Hacker-Mentalität geprüft. Die durchgeführten Tests sind meist manuelle Tests mit einem vergleichsweise niedrigen Potenzial zur Automatisierung, worauf im Verlauf noch näher eingegangen wird.

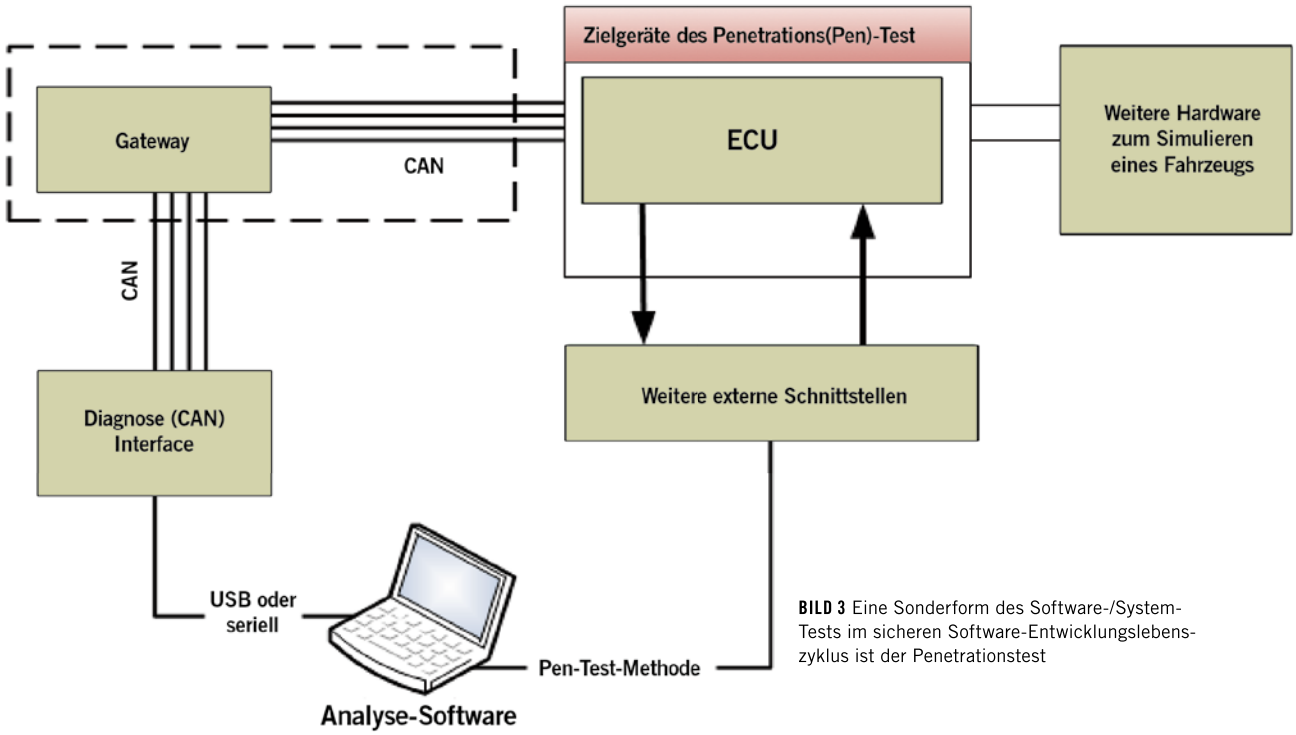



**BILD 3** Eine Sonderform des Software-/System-Tests im sicheren Software-Entwicklungslebenszyklus ist der Penetrationstest

## ROLLOUT UND BETRIEB

Für den Rollout beziehungsweise die Produktion muss in erster Linie sichergestellt werden, dass die Fertigung des Zielsystems mit sicheren Konfigurationseinstellungen erfolgt. In der Praxis bedeutet dies, dass alle entwicklungsspezifischen Vereinfachungen oder Sonderfunktionen abgeschaltet, entfernt oder zurückgebaut werden müssen.

Zudem muss zu diesem Zeitpunkt sichergestellt sein, dass alle Prozesse und Systeme in der Infrastruktur zur sicheren Übermittlung und Verarbeitung produktiver, sensibler Daten zwischen Entwicklung, Zulieferern und Produktion hinreichend abgesichert sind.

Für den Betrieb und den Produktlebenszyklus ist anschließend zum einen die Gewährleistung eines Feedbackkanals zur Entwicklung über festgestellte Sicherheitsprobleme sicherzustellen. Zum anderen muss die Risikoabschätzung für das Produkt regelmäßig neu durchgeführt werden.

Insbesondere im IT-Umfeld kann sich die Bedrohungslage kurzfristig ändern, beispielsweise wenn ein neuer Exploit oder eine grundsätzliche Schwachstelle in einem Verfahren veröffentlicht wird. Da aber gerade die Produkte im OEM-Umfeld eine sehr lange Zeit aktiv im Feld genutzt werden (zum Teil nach mehr als 20 Jahren), ist eine regelmäßige Neubewertung unumgänglich. Zudem müssen Maßnahmen etabliert werden, wie mit potenziellen Sicherheitslücken umgegangen werden soll beziehungsweise wie sie im Feld behoben werden können, sofern sich solche ergeben. Im V-Modell tangiert dies die Phase (7) Produktion & Betrieb.

## PENETRATIONSTESTS (PEN-TEST)

Als Sonderform des Software-/System-Tests im sicheren Software-Entwicklungslebenszyklus wurde bereits der Penetrationstest genannt. Hierbei handelt es sich um ein Testverfahren, **BILD 3**, das einen Angriff durch einen unberechtigten Dritten simuliert. Somit wird das System außerhalb seiner funktionalen Spezifikation auf Schwachstellen getestet. Je nach Informationsumfang, der zum Test zur Verfügung steht, lassen sich drei Testklassen unterscheiden:

- Black-Box: Es stehen keinerlei Informationen zu dem zu untersuchenden System zur Verfügung.
- Gray-Box: Es stehen teilweise Informationen, zum Beispiel Schnittstellenbeschreibungen, Architektur-/Design-Dokumente, zur Verfügung.
- White-Box: Die vollständige Dokumentation inklusive Source Code steht zur Verfügung.

Im V-Modell ist der Pen-Test normalerweise Teil der Phase (7) Operation, **BILD 3**. Im Steuergeräteumfeld sollten solche Tests aber bereits während der Entwicklung in den Phasen (4) bis (6) durchgeführt werden, da ein nachträgliches Patchen von Steuergeräten im Feld in der Regel problematisch oder zumindest kostenintensiv ist. Die Durchführung des eigentlichen Pen-Tests unterteilt sich schließlich in vier Phasen:

- Reconnaissance
- Enumeration
- Exploitation
- Dokumentation.

## RECONNAISSANCE

Diese Phase bezeichnet die aktive Informationsbeschaffung vor dem eigentlichen Pen-Test. Dabei werden entweder öffentlich nutzbare Informationsquellen verwendet, oder im Falle von White-Box-/ Gray-Box-Tests vom Auftraggeber bereitgestellte Dokumente gesichtet. In dieser Phase finden noch keine Angriffshandlungen statt. Ziel ist zunächst die Sammlung von Informationen zur Qualifikation geeigneter Einstiegspunkte für den Angriff (Phase 2: Enumeration).

## ENUMERATION

Jetzt werden geeignete Einstiegspunkte in das Zielobjekt identifiziert. Dazu erfolgt ein erstes aktives Ansprechen einzelner (Teil-)Systeme mit dem Ziel, weitere Informationen zu gewinnen oder offensichtliche Schwachstellen zu identifizieren. In diesen Zusammenhang wird ein Ranking der einzelnen Teilsysteme und Schnittstellen durchgeführt, um so das potenziell schwächste System zu ermitteln. Darüber hinaus erfolgen weitere vorbereitende Maßnahmen (weitere Informationsbeschaffung, Vorbereitung/Konfiguration des vorhandenen Toolings, projektspezifische Tool-Entwicklung).

## EXPLOITATION

Nun findet der eigentliche Angriff auf das System statt. Dazu werden im Vorfeld identifizierte Schwachstellen ausgenutzt und im Detail analysiert. Basierend auf diesen Erkenntnissen werden dann eigene oder bekannte verfügbare Exploits genutzt, um das System anzugreifen. Abhängig vom Zielsystem können hier verschiedene Methoden (Fuzzing, Scripting, DoS etc.) zum Einsatz kommen.

Im Automobilsteuergeräte-Umfeld ist zu beachten, dass sich die notwendigen Testumgebungen deutlich von denen für IT-Systeme unterscheiden. Während IT-Systeme in der Regel in ihrer echten Umgebung getestet werden, erfolgt ein Steuergerätetest oft an spezifischen Testsystemen (Teilsystemplätze) und eher selten im Serienfahrzeug. Zudem ist bei einem Steuergerät die Fehlererkennung – das heißt die Feststellung, dass ein Angriff ein Fehlverhalten verursacht hat – sehr zielsystemspezifisch, wohingegen IT-Komponenten wesentlich allgemeiner analysiert werden können.

## DOKUMENTATION

In dieser letzten Phase werden die Ergebnisse aus den vorangegangenen Phasen dokumentiert. Die Dokumentation sollte zumindest die folgenden Punkte beinhalten:

- Beschreibung identifizierter Schwachstellen (VIVA-Kriterien) mit Bewertung der Kritikalität
- Beschreibung und qualitative Bewertung des aus der Schwachstelle ableitbaren Risikos
- Darstellung geeigneter Maßnahmen zum Schließen der Schwachstelle, inklusive Zeitplan.

## ZUSAMMENFASSUNG

Zusammenfassend lässt sich sagen, dass ein sicheres Gesamtsystem nicht durch einzelne Sicherheitsfunktionen, Technologien oder ein einzelnes Kryptoverfahren sichergestellt werden kann. Und selbst gute Vorgaben liefern in der Regel noch genügend Freiräume zur falschen Interpretation, was gerade für den Bereich IT-Security kritisch ist. Hinzu kommt, dass IT-Security-Probleme oft durch Fehler in der Implementierung von Maßnahmen entstehen, die nicht mutwillig erfolgt sind. Solche Fehler lassen sich nicht kategorisch ausschließen, aber durch einen methodischen Ansatz sicher reduzieren. Wie aufgezeigt, gehören dazu im Wesentlichen:

- die saubere Analyse der Anforderungen, gerade vor dem Hintergrund des Datenschutzes und der Datensicherheit
- die Umsetzung eines durchgängig sicheren Software-Entwicklungslebenszyklus – zumindest für kritische Systeme
- die Verifikation und Absicherung von Systemen auch außerhalb ihrer funktionalen Spezifikation durch Pen-Tests.

Natürlich haben solche Lösungen Auswirkungen weit über die eigentliche Steuergeräteentwicklung hinaus. Somit müssen geeignete IT-Systeme für die Versorgung mit kryptografischen Daten und Diensten (PKI, Crypto Server, CAs, RAs) etabliert werden, die ihrerseits wiederum den Anforderungen aktueller Sicherheitsrichtlinien genügen (ISO27K, IT-Grundschutz etc.).

### LITERATURHINWEISE

**[1]** ISO 26262
**[2]** ETSI TS 102 165-1 V4.2.3 von 2011: Method and Proforma for Threat, Risk, Vulnerability Analysis:http://www.etsi.org/deliver/etsi_ts/102100_102199/10216501/04.02.03_60/ts_10216501v040203p.pdf