(Bild: Aktion Plagiarius)

Produktschutz und Lizenzierung:

Erst schützen und dann klagen

Der Rechtsweg ist der mühsamste Weg, um Produkte gegen Piraterie zu schützen. Viel effektiver ist es, das Kopieren so schwer wie möglich zu machen, um den zeitlichen Vorsprung eines Produkts so lange wie möglich zu halten.

Von Rüdiger Kügler

Zoll- und Polizeieinsatz auf der Messe: Fälschungen und Nachbauten beschlagnahmt! – Bei dieser Schlagzeile denkt man in der Regel zuerst an Designer-Jeans und Fußballtrikots aus Fernost, aber sie betrifft genauso Software und elektronische Geräte. Eine Zoll- und Polizeirazzia ist ein sehr medienwirksames Mittel gegen Raubkopierer und Nachahmer, hat jedoch leider meist nur einen kurzen und lokal begrenzten Erfolg. Und eine Klage kann sehr langwierig sein.

Erst vor wenigen Wochen führte ich eine Sicherheitsanalyse bei einem Hersteller von Equipment für Clubs und Diskotheken durch. Der Geschäftsführer erzählte mir, dass er seit einigen Jahren auf dem europäischen Markt nahezu exakte Kopien seiner Geräte findet – zu einem Bruchteil des Preises. Mittlerweile arbeitet er mit zwei europäischen Mitbewerbern zusammen, um solche Nachbauten auf Messen und Veranstaltungen beschlagnahmen zu lassen. Dies erschwert zwar den Eintritt in den europäischen Markt, hilft aber überhaupt nicht in Fernost. Dort wurden seinem offiziellen Händler sogar Nachbauten zum Weiterverkauf angeboten, frei nach dem Motto: „Lass' uns den Gewinn teilen und den Deutschen in die Röhre gucken."

Neben rein rechtlichen Möglichkeiten, von denen Zoll- und Polizeieinsätze nur eine Möglichkeit sind, gibt es technische Maßnahmen, um Raubkopien und Nachbauten zu erschweren und diese einfacher zu identifizieren. Eine Methode ist das Markieren der Originalware mit einem Wasserzeichen. Dies können Hologramme oder andere schwer zu kopierende Etiketten, Beimischungen in Plastikteilen oder stenographisch versteckte Merkmale in der Software sein. Aussicht auf Erfolg besteht natürlich nur, wenn der Markt, in dem die Geräte verkauft werden, auch einen rechtlichen Zugriff bietet. Gerade in Lateinamerika oder Fernost gestaltet sich das Klagen für Anbieter aus Deutschland eher schwierig.

Die gebräuchlichste Methode, um Raubkopien und Nachbauten zu erschweren, ist der Schutz des geistigen Eigentums durch Verschlüsselung und der Einsatz eines harten Kopierschutzes. Das Vervielfältigen der Software durch Duplizieren der SD-Karte oder eines anderen Speichermediums wird damit verhindert. Durch die Verschlüsselung wird das Reverse Engineering deutlich erschwert, mit der Folge, dass sich der Aufwand für den Nachahmer nicht mehr lohnt.

Neue Trends – Neue Risiken

Geräte bestehen in der Regel aus der eigentlichen Hardware und einem Softwareanteil. Der Softwareanteil wird meist als Firmware bezeichnet. In der Vergangenheit bestand die Hardware aus proprietären Komponenten wie ASICs oder speziell programmierten Mikrocontrollern. Es war nicht unüblich, die Software in einen ROM fest einzubrennen. Der Funktionsumfang der Geräte war auf die Bestimmung zugeschnitten. In den letzten Jahren konnte man die folgenden drei Trends beobachten:

- Hersteller gehen von proprietärer Hardware hin zu Standardkomponenten (**Bild 1**).
- Die Firmware der Geräte ist aktualisierbar, um neue Funktionen nachzuliefern.

➜ Immer mehr Funktionen werden durch die Software bestimmt und können auch nachträglich freigeschaltet werden.

Diese Trends verringern die Entwicklungs- und Produktionskosten, senken die Entwicklungszeiten und bieten neue Flexibilität.

Einsatz von Standardkomponenten

Durch die Verfügbarkeit von preiswerter Standardhardware und kostenfreien Betriebssystemen verschwindet proprietäre Hardware zusehends aus den Geräten. Einplatinencomputer, wie Raspberry Pi, Arduino, Galileo oder BeagleBoard, bieten eine kostengünstige Alternative zu eigener Hardware. Sie sind leistungsfähig und ohne eigenen Entwicklungsaufwand verfügbar. Als kostenfreie Betriebssysteme dienen vor allem Linux und Android. Damit werden lediglich ein Standardcompiler für C/C++ und eine passende Entwicklungsumgebung benötigt, und die Entwicklung beginnt.

Genauso schnell hat ein potenzieller Angreifer aber auch die Testumgebung für die Analyse der Software aufgebaut. Und Standarddecompiler können verwendet werden, um den Maschinencode in lesbaren Assemblercode zurück zu wandeln. Werkzeuge wie IDA Pro erzeugen dann sogar noch übersichtliche Ablaufpläne der Software. Früher musste der Angreifer Spezialwissen und oft teures Equipment besitzen, um analysieren zu können. Durch die Standardkomponenten reichen Profiwissen und kostengünstige Werkzeuge aus. Aber es kommt sogar noch schlimmer.

Der Einsatz von Java, vor allem auf Android, vereinfacht die Entwicklung noch weiter. Und durch .NET Core gibt es eine einfache Möglichkeit, .NET-Anwendungen auch auf Linux laufen zu lassen; dies vereinfacht die Entwicklung ebenso. Beide Lösungen, Java und .NET, bieten plattformunabhängige Entwicklung an. Damit wird die Portierung auf andere Systeme noch einfacher. Dabei gibt es einen intermediären Code, Java Byte Code und CIL (Common Intermediate Language), der sich sogar in Java oder eine .NET-Sprache zurückübersetzen lässt. Der potenzielle Angreifer erhält so quasi den Quellcode der Anwendung. Dies öffnet den Angreifermarkt für Hobby-Cracker.

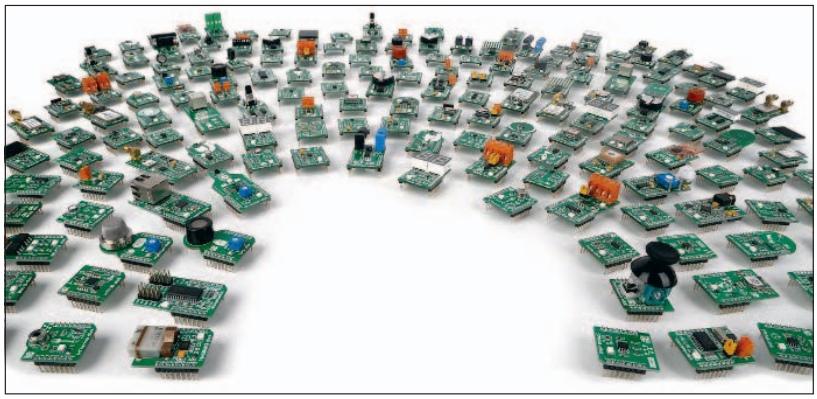

Bild 1. Für die Hardware werden immer mehr Standard-Komponenten verwendet. Diese große Vielfalt an „Click-Boards" kann z.B. über einen Adapter an Raspberry-Pi-Computer angeschlossen werden. (Bild: MikroE)

Aktualisierbare Firmware

Es gibt einige Beispiele, in denen Anbieter Firmware-Aktualisierungen nutzen, um das Produkt beim Kunden reifen zu lassen. So wurde die Festplattenaufnahmefunktion eines Sony-Fernsehers erst nach mehr als sechs Monaten nachgeliefert. Für den Hersteller des Gerätes ist dies ein großer Vorteil. Er kann Funktionen später nachliefern, wie in diesem Fall geschehen. Er kann aber auch Fehler beseitigen und so die Anzahl der Rückläufer und Garantiefälle minimieren.

Auch hier gibt es Ansatzpunkte für einen potenziellen Angreifer. Er könnte über die Aktualisierung eine manipulierte Firmware einspielen und damit die Kontrolle über Funktionen übernehmen, die nicht für ihn bestimmt waren. So könnten zum Beispiel Performance-Daten geändert werden, die mehr Leistung abrufen, aber das Gerät früher verschleißen lassen: der nächste vorprogrammierte Garantiefall.

Freischaltbare Softwarefunktionen

Als dritter Trend zeichnet sich ab, dass immer mehr Funktionen rein durch Software bestimmt werden. So sind aktuelle Steuerungen so leistungsfähig, dass diese mehrere Achsen gleichzeitig steuern können. Der Anwender ist aber bereit, für eine Steuerung mit drei Achsen mehr zu bezahlen als für eine Zwei-Achsen-Steuerung. Daher bestimmt die Software in diesem Beispiel, wie viele Achsen freigeschaltet werden. Dies hat viele Vorteile für Hersteller und Anbieter, vor allem, wenn diese Funktionen nachträglich freigeschaltet werden können. Der Hersteller fertigt weniger verschiedene Hardwaremodelle: eine Kostenersparnis in Produktion und Logistik. Der Anbieter kann dennoch für kleine Projekte die kostengünstige Zwei-Achsen-Version kaufen: eine Kostenoptimierung im Einkauf. Und das Beste ist, dass durch die nachträgliche Freischaltung Funktionen hinzugefügt werden können: Investitionssicherheit für den Anwender und Zusatzumsatz für den Hersteller.

Auch die Freischaltung von Funktionen ist ein beliebtes Angriffsziel von Crackern. Beinahe legendär ist der Fall, den mir ein Stickmaschinenhersteller erzählte. Er hatte mehrere Stickmaschinen nach Fernost verkauft, einmal die volle Ausbaustufe und mehrfach die kostengünstigste. Als sein Servicetechniker einen Einsatz vor Ort hatte, staunte er, dass alle Maschinen in der vollen Ausbaustufe liefen.

Gerade die Freischaltung und damit die Monetarisierung von Funktionen werden aktuell nachgefragt. Neben reinen Einmal-Aktivierungen führen erste Hersteller zeit- und verbrauchsbasierte Abrechnungsmodelle ein. Die Herausforderung hierbei sind die Geräte, die nicht permanent online sind.

Hardware als vertrauenswürdiger Anker

Das Freischalten von Funktionen erfolgt in den meisten Fällen durch eine Lizenzierung pro Gerät. Der Hersteller möchte verhindern, dass sein Anwender einmal eine Freischaltung kauft und diese auf allen Geräten einsetzt. Er möchte das Beispiel mit den Stickmaschinen ja nicht selbst erleben. Für die Umsetzung wird die Lizenz an einen sicheren Anker gebunden. Dies kann ein Dongle sein oder eine eindeutige und nur schwer fälschbare Eigenschaft des Gerätes. Prädesti-

niert für eine solche Eigenschaft sind ein Seriennummern-Chip oder ein TPM-Chip. Für Dongles gibt es weltweit mehrere Anbieter. Wibu-Systems bietet mit CodeMeter ein System, welches beides, Dongles und softwarebasierte Lizenzierung, unterstützt.

Von der Bindung an diesen Anker hängt die Sicherheit der Lizenzierung gegen Cracker ab. Die einfachste Art der Bindung ist das Auslesen einer Seriennummer und der Vergleich mit einem Wert in der Lizenz. In diesem Falle muss der Angreifer lediglich die Abfrage finden und einen bedingten Sprung in einen unbedingten Sprung oder nichts ändern. Beispiele sind JZ (Jump if Zero) und JNZ (Jump if not Zero) in einen JMP (unbedingter Sprung) oder nicht ändern (NOP). Allerdings bedeutet dies, dass der Anwender eine gepatchte Firmware benötigt.

Wahl der richtigen Implementierung

Schlimmer noch ist es, wenn der Cracker eine gültige Lizenz erzeugen kann. Dies ist die Höchststrafe für eine falsche Implementierung. Ich habe schon mehrfach eine Implementierung mit einer kryptographischen Prüfsumme (Hash) gesehen. In diesen Fall wurde beim Erzeugen der Lizenz die Eigenschaft mit den freigeschalteten Funktionen und einem geheimen Wert (Pepper) kombiniert und daraus ein Hash-Wert gebildet. Dieser Hash-Wert wurde in der Lizenz abgespeichert.

Beim Prüfen in der Software wird der gleiche geheime Wert (Pepper) wieder verwendet und aus der Eigenschaft und den freigeschalteten Funktionen der Hash-Wert berechnet. Dieses Verfahren hat eine große potenzielle Sicherheitslücke. Der Angreifer kann den geheimen Wert (Pepper) in der Software finden. Dann ist es ihm möglich, selber eine gültige Lizenz mit weiteren Funktionen oder für weitere Geräte zu erzeugen. Mit diesen kann dann die originale, nicht gepatchte Firmware verwendet werden.

Eine bessere Lösung ist die Verwendung von asymmetrischer Kryptographie. Die Lizenz wird mit dem privaten Schlüssel unterschrieben (signiert). Zum Prüfen benötigt die Software lediglich den zugehörigen öffentlichen Schlüssel. Aus diesem kann der private Schlüssel nicht berechnet werden. D. h. der Angreifer kann keine Lizenzen erzeugen, sondern muss jedes einzelne Gerät verändern.

Darüber hinaus gibt es Implementierungen, bei denen die Eigenschaft des Gerätes als privater Schlüssel verwendet wird. Der öffentliche Schlüssel wird dann zum Hersteller übertragen und der Hersteller verschlüsselt damit die Lizenz. Nur das originale Gerät kennt den privaten Schlüssel und kann die Lizenz dann entschlüsseln und verwenden. Selbiges gilt für gute Dongles. Hier werden die Lizenzen komplett im Dongle gespeichert und sind außerhalb nicht lesbar. Und damit sind sie auch nicht angreifbar.

Verschlüsselung von Code

Durch das Verschlüsseln und Unterschreiben der Lizenz kann verhindert werden, dass der Angreifer gültige

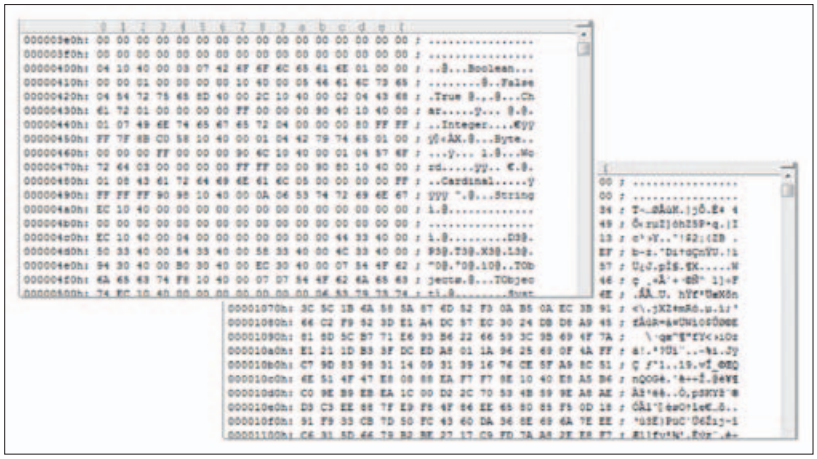

Bild 2. Für ein ausreichendes Schutzlevel muss der ausführbare Code (vorne) verschlüsselt werden (hinten). (Bild: Wibu-Systems)

Lizenzen erzeugt. In einigen Fällen ist dies ein ausreichender Schutz. Wenn man als Hersteller ein höheres Schutzlevel erreichen möchte, hilft die Option, den ausführbaren Code der Software zu verschlüsseln (**Bild 2**). Der Code wird vor der Auslieferung verschlüsselt. Entsprechende Werkzeuge wie zum Beispiel CodeMeter Protection Suite fügen einen eigenen Startcode zur verschlüsselten Software hinzu. Der Startcode überprüft beim Start der Software, ob eine gültige Lizenz vorhanden ist und die Software nicht verändert wurde (**Bild 3**). Im Fall von modularer Software mit mehreren Bibliotheken wird geprüft, ob diese ebenfalls unverändert sind. Erst dann wird der originale Code entschlüsselt und ausgeführt.

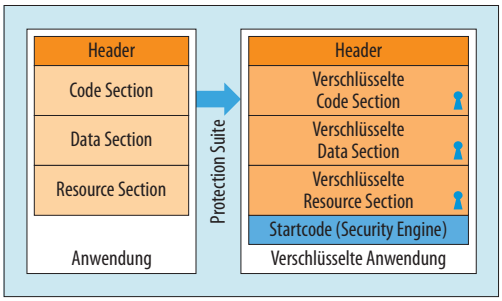

Bild 3. Werkzeuge wie die CodeMeter Protection Suite fügen einen eigenen Startcode zur komplett verschlüsselten Software hinzu. Der Startcode überprüft beim Start der Software, ob eine gültige Lizenz vorhanden ist und die Software nicht verändert wurde. (Quelle: Wibu-Systems)

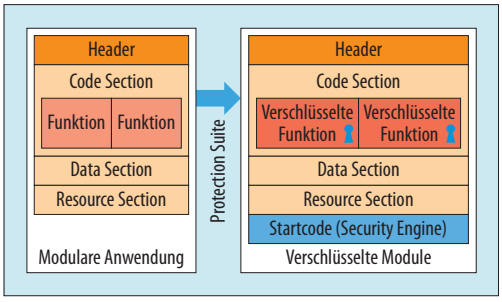

Bild 4. Wenn bestimmte Teile des Codes unverschlüsselt bleiben sollen, weil sie immer ausführbar sein müssen, bietet sich die Verschlüsselung einzelner Module an. (Quelle: Wibu-Systems)

Ein wesentlicher Punkt für die Sicherheit ist es, ob die Lizenz zum Entschlüsseln des Codes benötigt wird oder ob dies unabhängig davon erfolgt. Durch eine Bindung an die Lizenz wird die Gruppe der möglichen Angreifer auf Besitzer einer Lizenz eingeschränkt. Durch Gegenmaßnahmen wie das Sperren der Lizenz bei erkannten Attacken kann man die Anzahl der Angriffsversuche beschränken. Hier liegt ein großer Vorteil von Dongles als sicherer Speicher, da das Sperren persistent gesetzt werden kann. Die Entkopplung von Verschlüsselung und Lizenzierung hat allerdings auch ihre Berechtigung, vor allem dann, wenn aus Safety-Gründen Grundfunktionen immer ausgeführt werden müssen.

Modulare Verschlüsselung

Im Bereich Safety gibt es Funktionen, die immer ausführbar sein müssen. Andere Funktionen sollen hart gegen Reverse Engineering und Raubkopien geschützt werden. Diese Anforderung trifft natürlich nicht nur auf Safety-Anwendungen zu, sondern auf alle Geräte, bei denen einzelne Funktionen lizenziert werden sollen. Die optimale Lösung ist hier:

- ➜ Man definiere verschiedene Lizenzen für die einzelnen Funktionen. Optimalerweise hat jede einzelne Lizenz einen anderen kryptographischen Schlüssel.
- ➜ Man identifiziere den Teil des ausführbaren Codes, der genau dieser Funktionalität zugeordnet ist. Im Falle der verschiedenen Achsen dürfte dies eher schwierig sein. Anders ist das bei einer Funktion, die etwas optimiert, zum Beispiel den Verbrauch von Material. Hierfür sollte die Zuordnung recht einfach sein.
- ➜ Man verschlüsselt diesen Teil des ausführbaren Codes mit dem Schlüssel der zu dieser Funktion gehörigen Lizenz (**Bild 4**).

Nun ist eine Entschlüsselung des Codes nur möglich, wenn die zu dieser Funktion passende Lizenz mit diesem kryptographischen Schlüssel auch verfügbar ist; wieder eine Einschränkung des Kreises der Angreifer.

Schutz durch „Blurry Box"

Blurry Box ist ein vom FZI (Forschungszentrum Informatik), KIT (Karlsruhe Institute of Technology) und Wibu-Systems entwickeltes Verfahren, mit dem der Aufwand für den Angreifer so hoch wird wie die neue Implementierung des kompletten Codes. Blurry Box kombiniert dabei verschiedene Methoden. Bestandteile sind unter anderem:

- ➜ die bereits genannte Verschlüsselung von Teilen des ausführbaren Codes,
- ➜ Erzeugen von verschiedenen Varianten für den gleichen Code,
- ➜ Einfügen von Fallen, welche die Lizenz sperren,
- ➜ Auslagerung von Entscheidungen an einen Dongle,
- ➜ Erkennung von korrekten Zuständen beim Durchlaufen des Codes,
- ➜ Geschwindigkeitsbremse beim Entschlüsseln, um Brute-Force-Attacken zu erschweren.

Code in einen Dongle auslagern

Im Rahmen von Blurry Box hat sich auch Wibu-Systems mit dem Thema beschäftigt, wie und welchen Code man im Dongle ausführen kann. Moderne Dongles enthalten einen Mikrocontroller, bei Wibu den SLE97 von Infineon, und können Code im Dongle ausführen. Die große Herausforderung ist die Auswahl des Codes, der im Dongle ausgeführt werden soll. Ist der Code zu trivial, dann kann er nachgebildet werden. Ist der Code zu komplex, hat dies zu große Auswirkungen auf die Performance.

Bild 5. Die Codemeter-Hardware stellt den vertrauenswürdigen Anker für die Software dar. Mit dem Blurry-Box-Verfahren können Teile der Software auch auf einen Dongle ausgelagert werden. (Bild: Wibu-Systems)

Bei Blurry Box wird Code in den Dongle ausgelagert, der für das Blurry-Box-Verfahren notwendig ist. Prinzipiell ist es aber auch möglich, selber Code zu wählen und diesen auszulagern. Dies wird auch als CodeMoving bezeichnet.

Im Falle von CodeMeter (**Bild 5**) können viele Funktionen ausgelagert werden. Diese Funktionen werden in ANSI-C geschrieben. Dabei kann eine Funktion weitere Unterfunktionen enthalten. Diese werden ebenfalls mit ausgelagert. Im Wesentlichen sind folgende Rahmenbedingungen zu beachten:

- ANSI-C Code (der Code wird für den Dongle neu kompiliert),
- keine Aufrufe von externen Bibliotheken oder Funktionen (diese stehen im Dongle nicht zur Verfügung),
- maximale Größe von 3 KByte für Code und Daten (der Dongle hat einen begrenzten Speicher).

Beim Verschlüsseln der Software wird zusätzlich der Source Code für die auszulagernden Teile angegeben. Diese Teile werden von der CodeMeter Protection Suite (AxProtector) neu kompiliert und verschlüsselt. Die verschlüsselten Teile werden als Binärdaten in der Anwendung abgelegt.

Beim Ausführen der Software werden die verschlüsselten Binärdaten und die Eingabeparameter an den Dongle übergeben, sobald die Funktion benötigt wird. Die Binärdaten werden dann im Dongle entschlüsselt und mit den übergebenen Eingabeparametern ausgeführt. Um Werte zwischen Funktionsaufrufen zu behalten, steht im Dongle ein Zustandsspeicher zur Verfügung. Als Ergebnis erhält die Anwendung die Ausgabeparameter zurück.

Aktuell wird diese Technologie in Bereichen eingesetzt, in denen die Gefahr von Raubkopien und Nachbauten besonders hoch ist. Ein Beispiel ist das eingangs erwähnte Equipment für Diskotheken und Clubs. Die Integration dieser Technologie erfordert einen entsprechenden Aufwand der Hersteller, die Funktionen auszuwählen und ggf. nach ANSI-C portieren müssen. In der Regel ist eine direkte Unterstützung durch den Anbieter der Technologie sinnvoll.

Signatur beweist Authentizität der Software

Ein letztes Wort zu Firmware-Aktualisierungen. Das empfohlene Vorgehen ist die Signatur der Firmware mit einem privaten Schlüssel durch den Hersteller. Im Gerät selber sollte der Aktualisierungsmechanismus aus zwei Teilen bestehen: der eigentlichen Firmware und einem Lademechanismus. Optimalerweise ist der Lademechanismus klein und nicht änderbar. Er enthält den öffentlichen Schlüssel, um die Signatur der Firmware zu überprüfen und erlaubt die Aktualisierung nur, wenn diese stimmt.

Verschlüsselung gegen Reverse Engineering und Raubkopien sowie eine sichere Firmware-Aktualisierung können miteinander kombiniert werden, wenn beides benötigt wird. Diese technischen Maßnahmen können bei richtigem Einsatz die Erstellung von Raubkopien und Nachbauten erschweren und verhindern. Der Hersteller kann damit seinen Vorsprung länger wahren und die Anzahl der Zoll- und Polizeieinsätze auf der Messe und die Klagen können reduziert werden. *jk*

Rüdiger Kügler
arbeitet seit 2003 als Security Expert bei Wibu-Systems und leitet das Professional Services Team. Seine Kernkompetenzen sind der Schutz von Software gegen Reverse Engineering, die Integration von Lizenzierung in die internen Geschäftsprozesse von Softwareherstellern und die Optimierung der Verteilungs- und Verwaltungsprozesse für Software und Lizenzen.
ruediger.kuegler@wibu.com