Fraunhofer SIT

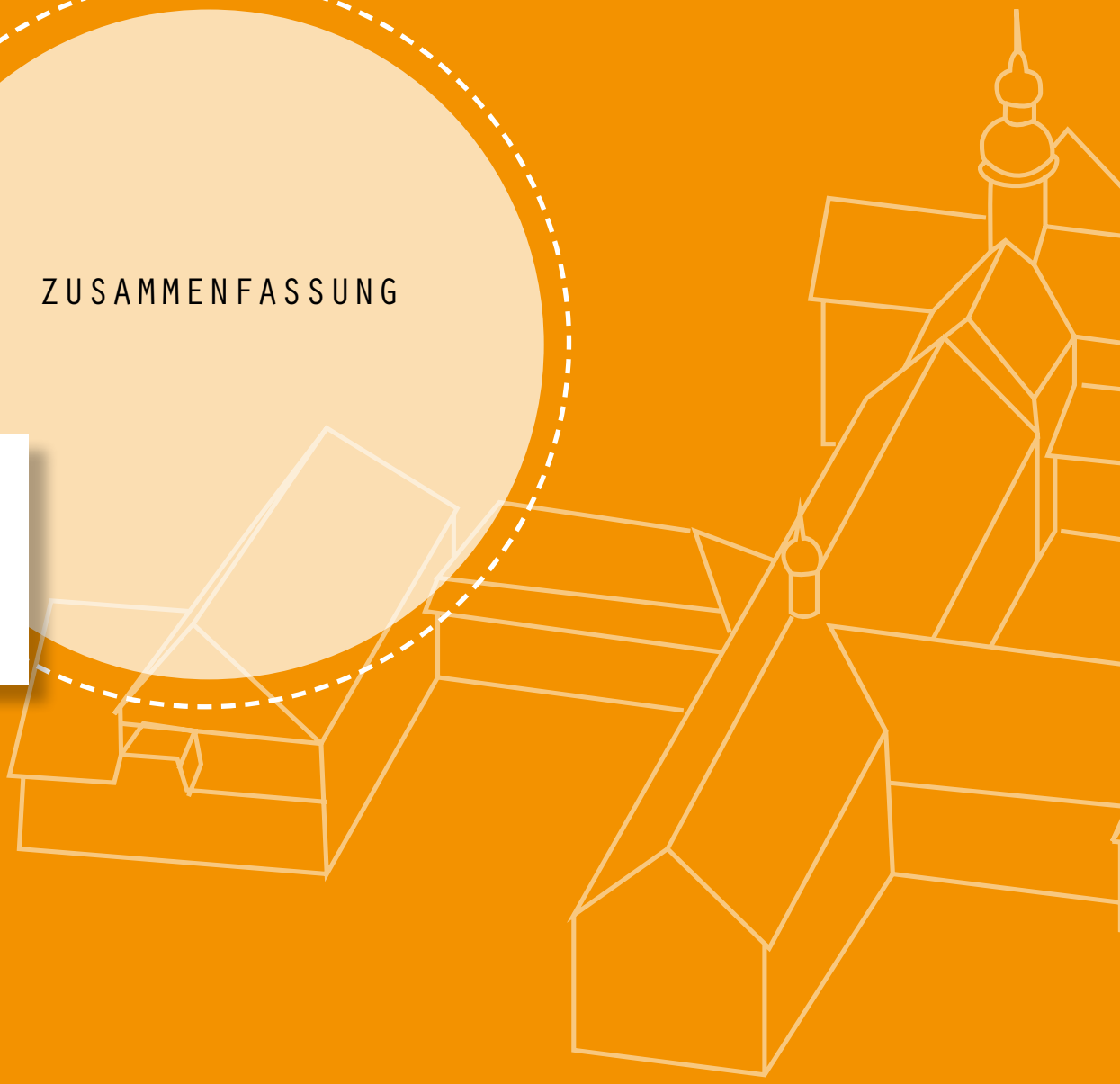
FRAUNHOFER-INSTITUT FÜR SICHERE INFORMATIONSTECHNOLOGIE

EBERBACHER GESPRÄCH »SICHERE SOFTWAREENTWICKLUNG«

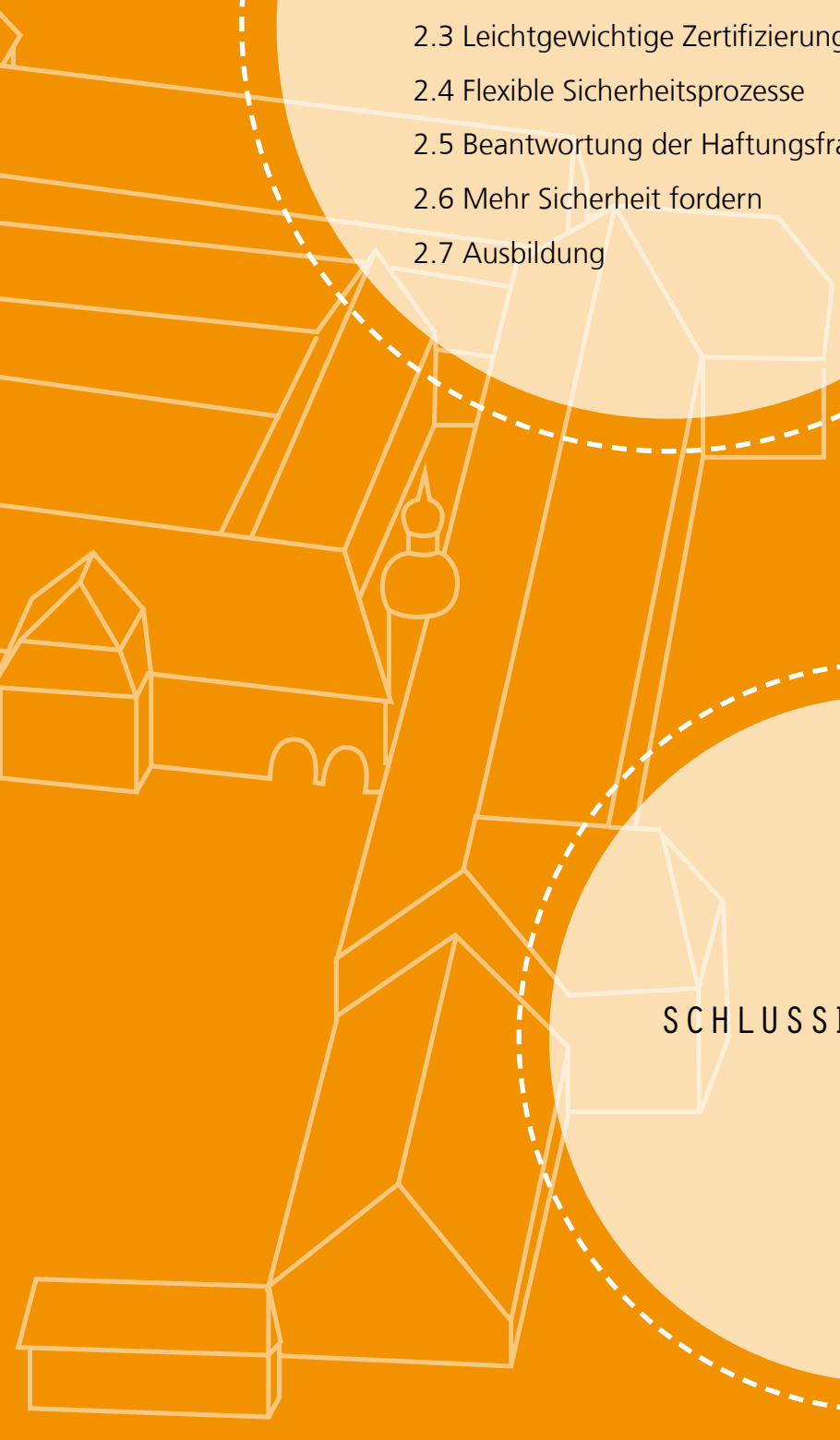
Eberbacher Gespräche

ZUSAMMENFASSUNG

1



INHALT



2 HERAUSFORDERUNGEN UND HANDLUNGSEMPFEHLUNGEN

2.1 Automatisierte Testwerkzeuge

2.2 Messbarkeit von Software-Sicherheit

2.3 Leichtgewichtige Zertifizierung

2.4 Flexible Sicherheitsprozesse

2.5 Beantwortung der Haftungsfrage

2.6 Mehr Sicherheit fordern

2.7 Ausbildung

3 SCHLUSSBETRACHTUNG

VORWORT

Was sind die aktuellen Herausforderungen für IT-Sicherheit und Datenschutz in der Praxis? Was ist in der Zukunft zu erwarten? Was kann und soll Technik leisten, wo sind die Grenzen des Machbaren? Wo braucht es neue Ideen? Angewandte Forschung braucht den Dialog zwischen Wissenschaft und Wirtschaft, um Antworten auf solche grundsätzlichen Fragen zu erhalten. Die »Eberbacher Gespräche« des Fraunhofer SIT bieten ein Forum für diesen Dialog. Experten aus Wissenschaft und Wirtschaft treffen sich für einen Tag im Kloster Eberbach und erarbeiten zu einem Thema gemeinsam Antworten auf diese Fragen. Im Juni 2013 war das Thema »Sichere Softwareentwicklung«.

Teilnehmer waren

Prof. Dr. Eric Bodden	Fraunhofer SIT/TU Darmstadt
Roman Drahtmüller	SUSE Linux Products GmbH
Dr. Johann Fichtner	Siemens AG
Dr. Thorsten Henkel	Fraunhofer SIT
Dr. Ingo Hollenbeck	Robert Bosch GmbH
Gerold Hübner	SAP AG
Dr. Michael Kreutzer	TU Darmstadt
Oliver Küch	Fraunhofer SIT
Prof. Dr. Igor Podebrad	Commerzbank AG
Dr. Markus Schneider	Fraunhofer SIT
Dr. Harald Schöning	Software AG
Prof. Dr. Gerald Spindler	Georg-August-Universität Göttingen
Sven Türpe	Fraunhofer SIT
Prof. Dr. Michael Waidner	Fraunhofer SIT/TU Darmstadt

Die in diesem Papier dargestellten Ergebnisse werden von den Teilnehmern unterstützt, stellen aber nicht notwendigerweise die Sichtweise des jeweiligen Arbeitgebers dar.

1 ZUSAMMENFASSUNG

Sichere Software ist grundlegend für jegliches Bemühen um IT-Sicherheit. Vermutlich enthält jede größere Software Schwachstellen, die Angreifer ausnutzen können. Viele IT-Sicherheitsexperten sehen dies als größte Gefahrenquelle in der IT. [1]

Gegen Software-Schwachstellen können sich die Anwender selbst kaum schützen. Firewalls, Viren-Scanner und spezielle Anwendungs-Wrapper helfen zwar, Angreifer auf Abstand zu halten, schützen aber meist nicht vor der Ausnutzung unbekannter Schwachstellen. [2] Zwar können Experten theoretisch Lücken im Programmcode der schadhaften Anwendung selbst schließen. Dies steht jedoch zumindest bei Closed-Source-Software rechtlich im Gegensatz zum Urheberrecht des Softwareherstellers. [3] Die Hauptverantwortung für sichere Software liegt dementsprechend bei den Softwareherstellern – die oft nicht in der Lage zu sein scheinen, geeignete Schritte zur Verbesserung der Sicherheit ihrer Produkte zu unternehmen.

Um die Probleme und Beweggründe der Softwarehersteller genauer zu verstehen und Handlungsmöglichkeiten für Unternehmen, Politik und Gesellschaft zu erarbeiten, veranstaltete das Fraunhofer-Institut für Sichere Informationstechnologie SIT am 4. Juni 2013 das Eberbacher Gespräch zum Thema »Sichere Softwareentwicklung«. Im Rahmen dieses Gesprächs identifizierten die Teilnehmer sieben Problemfelder und sammelten konkrete Empfehlungen, mit denen sich die Entwicklung sicherer Software nachhaltig fördern lässt:

1. Automatisierte Testwerkzeuge
Heutige kommerziell verwendete Testwerkzeuge erfüllen die Anforderungen von Unternehmen nur unzureichend. Gerade in Deutschland wird an diesem für die Wirtschaft wichtigen Problem intensiv und erfolgreich geforscht. Die Politik sollte diesen Umstand nutzen und den Wissenstransfer in den wirtschaftlichen Alltag aktiv fördern und gestalten.

2. Messbarkeit von Software-Sicherheit
Damit Unternehmen ihre Investitionen zielgerichtet steuern können, müssen Änderungen im Sicherheits- bzw. Risikoniveau einzelner Produkte möglichst genau und quantitativ darstellbar und vorhersagbar sein. Wirtschaft und Forschung sollten zusammen entsprechende allgemein akzeptierte quantitative Modelle und Methoden schaffen.

3. Leichtgewichtige Zertifizierung
Bestehende Zertifizierungsschemata sind als Sicherheitsnachweis in den meisten Fälle ungeeignet, weil sie in ihrer Aussagekraft sehr begrenzt und fehleranfällig sind. Bevor ein neues praxistaugliches Zertifizierungsschema für Software entwickelt wird, sollten Wirtschaft und Forschung zunächst gemeinsam ausführlich die Defizite bestehender Verfahren analysieren. Anschließend gilt es, eine aussagekräftige, leichtgewichtige und änderungsfreundliche Zertifizierung von Software zu entwickeln, die zum Beispiel auch die Versionsentwicklung einer Software berücksichtigt.

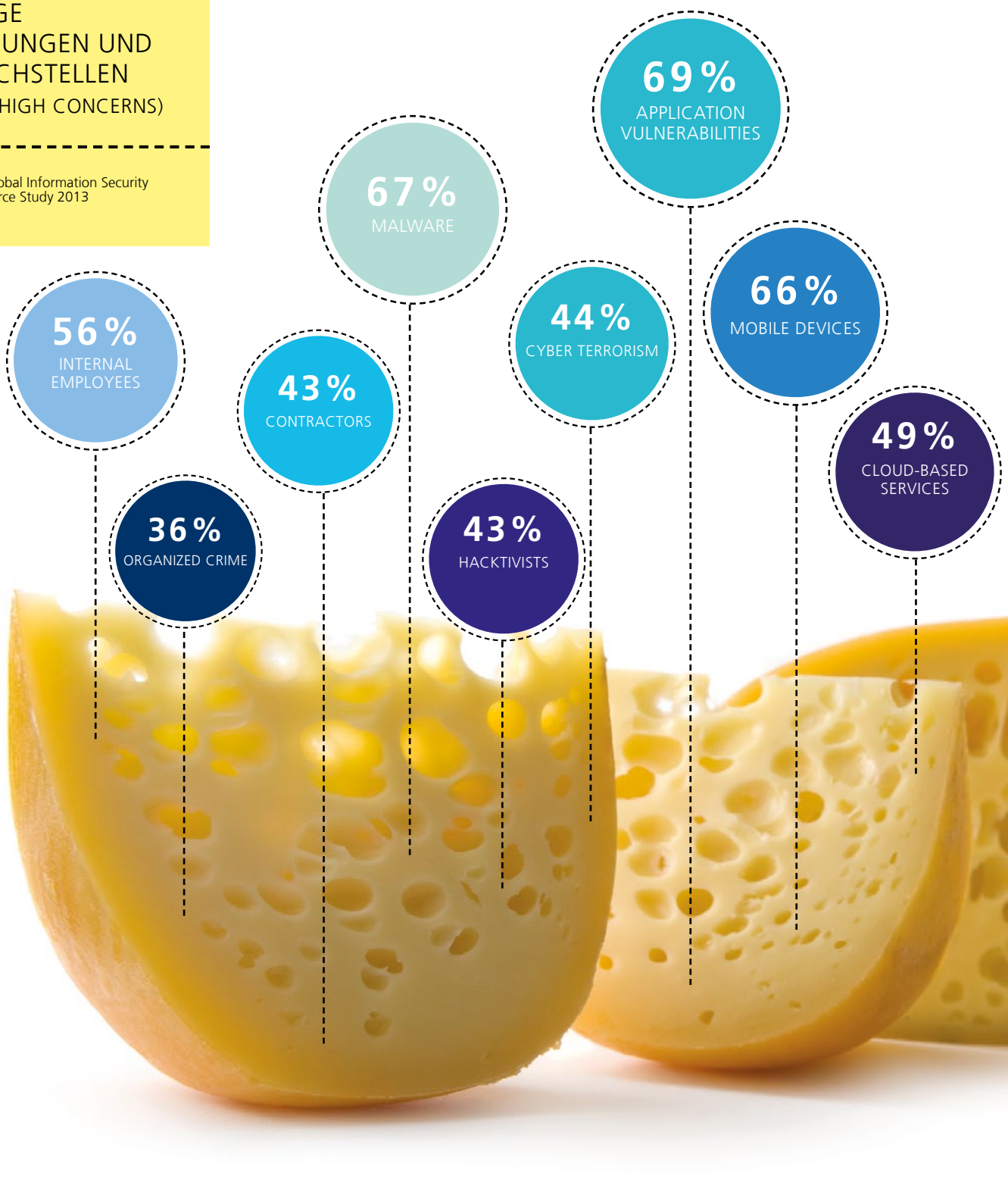
1 The 2013 (ISC)2 Global Information Security Workforce Study

2 Michael Waidner, Michael Backes, Jörn Müller-Quade (Hrsg.): Entwicklung sicherer Software durch Security by Design; SIT Technical Report, Fraunhofer Verlag, München, 2013;
https://www.sit.fraunhofer.de/fileadmin/dokumente/studien_und_technical_reports/Trendbericht_Security_by_Design.pdf

3 Eric Bodden, Siegfried Rasthofer, Philipp Richter, Alexander Roßnagel: Schutzmaßnahmen gegen datenschutz-unfreundliche Smartphone-Apps, In: Datenschutz und Datensicherheit, 2013

WICHTIGE BEDROHUNGEN UND SCHWACHSTELLEN
(TOP AND HIGH CONCERNS)

Quelle: (ISC)² Global Information Security Workforce Study 2013



1 ZUSAMMENFASSUNG

4. Flexible Sicherheitsprozesse
Während die meisten großen global agierenden Hersteller dem Thema IT-Sicherheit bereits einen großen Stellenwert einräumen, verwendet ein Großteil der kleinen und mittleren Software-Hersteller immer noch keine klar definierten Prozesse zur Entwicklung sicherer Software. Diese Unternehmen scheuen oft den hohen Einführungsaufwand und befürchten die Störung ihrer agilen Entwicklungsmethoden. Wirtschaft und Forschung sollten gemeinsam einen Baukasten für Sicherheitsprozesse entwickeln, der es Herstellern und Entwicklungsteams leicht macht, den für sie optimalen Prozess zusammenzustellen und zu etablieren.

5. Beantwortung der Haftungsfrage
Wer haftet eigentlich für Schäden durch Software-Schwachstellen? Zur Beantwortung der Frage erscheint es sinnvoll, eine Diskussionsplattform für Gesetzgeber, Softwarehersteller und Wissenschaftler aus Recht und Technik zu schaffen. Eine Haftungsklärung könnte sich positiv auf das allgemeine IT-Sicherheits- und Datenschutzniveau auswirken und zu einem Wettbewerbsvorteil für deutsche IT-Hersteller führen.

6. Mehr Sicherheit fordern
Um die Nachfrage nach sicherer Software zu fördern, sollten staatliche Vergaberichtlinien so verändert, dass Mindestanforderungen eingehalten werden. Darüber hinaus sollten auch finanzielle Anreize für entsprechende Forschungsförderung geschaffen werden. Innovative Versicherungsmodelle bilden einen weiteren vielversprechenden Ansatzpunkt.

7. Ausbildung
Die Komplexität von Software und deren Sicherheit kann nur noch in Teams von sehr gut ausgebildeten Personen bewältigt werden. Fachleute mit entsprechend breiten Qualifikationen und tiefem IT-Sicherheitsverständnis sind selten. Deshalb braucht es verstärkte Bemühungen in allen Bereichen der IT-Ausbildung.

2 BESTANDSAUFNAHME UND HERAUSFORDERUNGEN

2.1 AUTOMATISIERTE TESTWERKZEUGE

Heutige Werkzeuge für Softwaresicherheit steigern die Sicherheit des Endprodukts bereits effektiv in allen Entwicklungsphasen. Große Softwarehersteller setzen sie bereits in allen Phasen der Softwareentwicklung ein. Leider gibt es große Hemmnisse für deren Einsatz bei kleinen und mittleren Software-Anbietern. Hier fehlt zum Beispiel das erforderliche Expertenwissen für den Einsatz, und der Einarbeitungsaufwand ist oft sehr hoch. Ein Problem, das auch größere Kunden plagt, ist die Tatsache, dass die Werkzeuge oft zu viele »False Positives« melden (low precision), jedoch viele sicherheitsrelevante Fehler bei Codeanalysen oft nicht entdecken (low recall). Für beide Messgrößen, Precision und Recall, gibt es derzeit in der Industrie keine anerkannten Messverfahren, die es Nutzern erlauben, die Qualität der Werkzeuge zu vergleichen. Ebenso fehlt es an Möglichkeiten festzustellen, ob sich Werkzeuge mit unterschiedlicher Qualität oder unterschiedlichem Fokus sinnvoll ergänzen lassen. Hinzu kommt, dass viele Tools Produktlinien nicht unterstützen, die Freiheitsgrade der Entwicklungsteams einschränken und sich nur aufwendig in die bestehende Infrastruktur integrieren lassen. Einen weiteren Hinderungsgrund stellen die für KMUs oft sehr hohen Anschaffungskosten dar. Kurzfristig werden die Werkzeuge Experten vorbehalten bleiben. Die Nutzbarkeit von Werkzeugen für Nicht-Sicherheitsexperten wird angesichts der inhärent hohen Komplexität des Problemfeldes als nur mittelfristig zu erreichendes Ziel angesehen. Hier ist erheblicher Forschungsaufwand zu leisten, bis ein Durchbruch erreicht werden kann.

Vollständigkeit vs. Relevanz

Werkzeuge zur Codeanalyse, die aus der Forschung kommen, streben meist Vollständigkeit an. Dies entspricht jedoch nicht den Anforderungen der Unternehmen. So haben Forschungswerkzeuge meist das Ziel, bestimmte Fehlertypen in einer Software vollständig aufzulisten (Perfect Recall). Zielsetzung ist also, keinen Fehler zu übersehen und Zero False Negatives zu erreichen. Gleichzeitig wird akzeptiert, dass sich viele der aufgelisteten Fehler im Nachhinein als unschädlich (False Positives) erweisen. Diese Zielsetzung entspricht dem Selbstverständnis der Forschung, macht aber die meisten dieser Werkzeuge für den Alltagseinsatz untauglich. In der Praxis ist es für einen Entwickler nicht möglich, z. B. 1000 gemeldete Fehler zu analysieren mit dem Wissen, dass sich darunter vermutlich 999 »False Positives« und ein echter Fehler verbergen.

Kommerzielle Werkzeughersteller filtern deswegen mehr oder weniger großzügig gefundene Fehler aus und lassen den Anspruch der Vollständigkeit zugunsten der Relevanz fallen. Die Abwägung zwischen Relevanz und Vollständigkeit wird stets eine Gratwanderung bleiben, denn entsprechend dem Theorem von Rice sind Programmanalyseprobleme generell nicht eindeutig zu entscheiden. Dies allerdings verhindert nicht, die bestmöglichen heuristischen Lösungen in der angewandten Forschung zu suchen und zu finden. Auch in diesem Bereich besteht also hoher lohnenswerter Forschungsbedarf. Glücklicherweise hat sich in den letzten Jahren die Zielorientierung der Forschung langsam in Richtung der Marktbedürfnisse verändert. [1]

Unterstützung von Produktlinien

Software-Produktlinien sind ein gängiger Ansatz der Software-Industrie, mittels Wiederverwendung von Programmcode Kosten zu senken. Bei einer Software-Produktlinie gibt es eine Code-Basis, die den Code der gesamten Produktlinie enthält. Die Produktlinie kennt eine Anzahl an üblicherweise optionalen Funktionen (Features), und die eigentlichen Produkte entstehen durch das An- und Abwählen derselben. Bei n optionalen Features können so bis zu 2^n verschiedene Produkte aus einer einzigen Code-Basis entstehen. Bis vor einem Jahr waren Software-Produktlinien deshalb kaum mit plausiblem Aufwand mittels automatisierter Werkzeuge analysierbar, da letztlich

1 Ben Livshits, Manu Sridharan, Yannis Smaragdakis, and Ondřej Lhoták: In Defense of Unsoundness; http://soundiness.org/

ANZAHL DER SCHWACHSTELLEN IN DEN 50 MEIST GENUTZTEN PROGRAMMEN

Quelle: Secunia Vulnerability Review 2013

DIE TOP 10

Google Chrome	291
Mozilla Firefox	257
Apple ITunes	243
Adobe Flash Player	67
Oracle Java JRE SE	66
Adobe AIR	56
Microsoft Windows 7	50
Adobe Reader	43
Microsoft Internet Explorer	41
Apple Quicktime	29

63
SCHWACHSTELLEN PRO PRODUKT

EINE STEIGERUNG VON 98% IN 5 JAHREN

2 HERAUSFORDERUNGEN UND EMPFEHLUNGEN

jede der exponentiell vielen Feature-Kombinationen getrennt analysiert werden musste. Prof. Eric Bodden et. al. erfand ein neues Verfahren, Produktlinien effizient und effektiv für die automatisierte Analyse zugängig zu machen.[2] Der exponentielle Blowup bleibt hierbei in der Praxis aus. Schwachstellen lassen sich in allen möglichen Feature-Kombinationen gleichzeitig früh erkennen. Die aktuelle Herausforderung ist, dieses vielversprechende Ergebnis in die Praxis zu überführen.

Der Werkzeugeinsatz darf nur minimal zur Einschränkung der Freiheitsgrade für die Entwicklungsteams führen. Selbstverständlich ist es für Programmierteams eine begrüßenswerte Maxime, dass die Freiheitsgrade einer Programmiersprache nicht durch nachgelagerte Werkzeuge eingeschränkt werden, gleichzeitig könnte diese Maxime starke Analysemethoden verhindern. Hier den guten Mittelweg für alle Seiten zu finden, ist eine weitere lohnenswerte Anstrengung.

Zu guter Letzt stehen Anschaffungskosten und Glaubwürdigkeit auf dem Prüfstand: Den Kauf der wirklich effektiven Flaggschiff-Werkzeuge für Softwaresicherheit aus den USA leisten sich derzeit vorwiegend große Unternehmen. Gleichzeitig stehen Unternehmen aus den USA seit den Enthüllungen von Edward Snowden oft unter Verdacht, die technischen Möglichkeiten absichtlich nicht vollständig auszunutzen. Da auch Nachrichtendienste Software-Schwachstellen zur Informationsgewinnung ausnutzen können, haben diese staatlichen Dienste auch Interesse daran, dass gewisse Klassen von Schwachstellen von Testwerkzeugen nicht gefunden werden. Eine stärkere nationale Diversifizierung der Anbieter / Hersteller scheint deshalb wünschenswert.

Ein weiteres Problem stellen Analysedienste dar, die ihre Leistungen als Cloudservice anbieten. Auch deutsche Softwarehersteller schicken in regelmäßigen Abständen ihren Programmcode in eine in den USA lokalisierte Cloud-Umgebung, um den Code auf Schwachstellen analysieren zu lassen. Es ist bei der derzeitigen Rechtslage in den USA nicht auszuschließen, dass Geheimdienste und andere Zugriff auf diese Schwachstellen erlangen und somit eine exklusive Möglichkeit haben, diese frühzeitig für politische und wirtschaftliche Zwecke auszunutzen.

Handlungsempfehlungen

Aus der Sicht der Gesprächsteilnehmer wäre eine große europäische Anstrengung hin zu kostengünstigeren Softwaresicherheitswerkzeugen mit einem tragfähigen ökonomischen Modell sehr lohnenswert. Diese europäische Forschungs- und Entwicklungsanstrengung kann nicht allein von der Industrie geleistet werden.[3] Die Teilnehmer fordern deshalb die Beteiligung der nationalen und europäischen Wirtschafts- und Forschungspolitik an dieser gemeinsamen Anstrengung. Zukünftige Werkzeuge sollten langfristig auch für Programmierer einsetzbar sein, die keine Sicherheitsexperten sind. Der Einarbeitungsaufwand sollte gering sein, die Werkzeuge sollten False Positives minimieren und gleichzeitig möglichst viele sicherheitsrelevante Fehler finden (high precision and recall). Die Werkzeuge sollten Produktlinienanalyse unterstützen, die Freiheitsgrade für die Entwicklungsteams bestmöglich aufrechterhalten und sich leicht in die bestehende Infrastruktur integrieren lassen. Des Weiteren sollten Standards entwickelt werden, um die absolute und relative Güte von Analysewerkzeugen bestimmen zu können.

2 Eric Bodden, Társis Tolêdo, Márcio Ribeiro, Claus Brabrand, Paulo Borba, Mira Mezini; SPLLIFT: statically analyzing software product lines in minutes instead of years. In Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation (PLDI ,13). ACM, New York, NY, USA, 2013; 355-364. DOI=10.1145/2491956.2491976 http://doi.acm.org/10.1145/2491956.2491976

3 M. Waidner, M. Backes, J. Müller-Quade; Positionspapier Sicherheitstechnik im IT-Bereich, Darmstadt 2013; https://www.sit.fraunhofer.de/fileadmin/dokumente/studien_und_technical_reports/Positionspapier_IT-Sicherheit_Forschung.pdf

DAS KUCKUCKSEI IM EIGENEN NEST – ANTEIL DER SCHWACH-STELLEN, DIE IN FREMD-SOFTWARE ENTDECKT WERDEN

Quelle: Secunia Vulnerability Review 2013

2 HERAUSFORDERUNGEN UND EMPFEHLUNGEN

2.2 MESSBARKEIT VON SOFTWARESICHERHEIT

So wünschenswert und nützlich die Messbarkeit von Softwaresicherheit aus naheliegenden Gründen ist, umso verwunderlicher scheint es, dass es noch keine etablierte und anerkannte Methode hierfür gibt. Angesichts der Komplexität wird dies jedoch offensichtlich, denn große Software-Anwendungen gehören zu den komplexesten menschgemachten Artefakten. Rein statistisch enthalten große Softwarepakete hunderte Sicherheitsprobleme, die bei Auslieferung unentdeckt sind. Selbstverständlich jedoch sind nicht alle diese Sicherheitsprobleme als gravierend einzustufen.

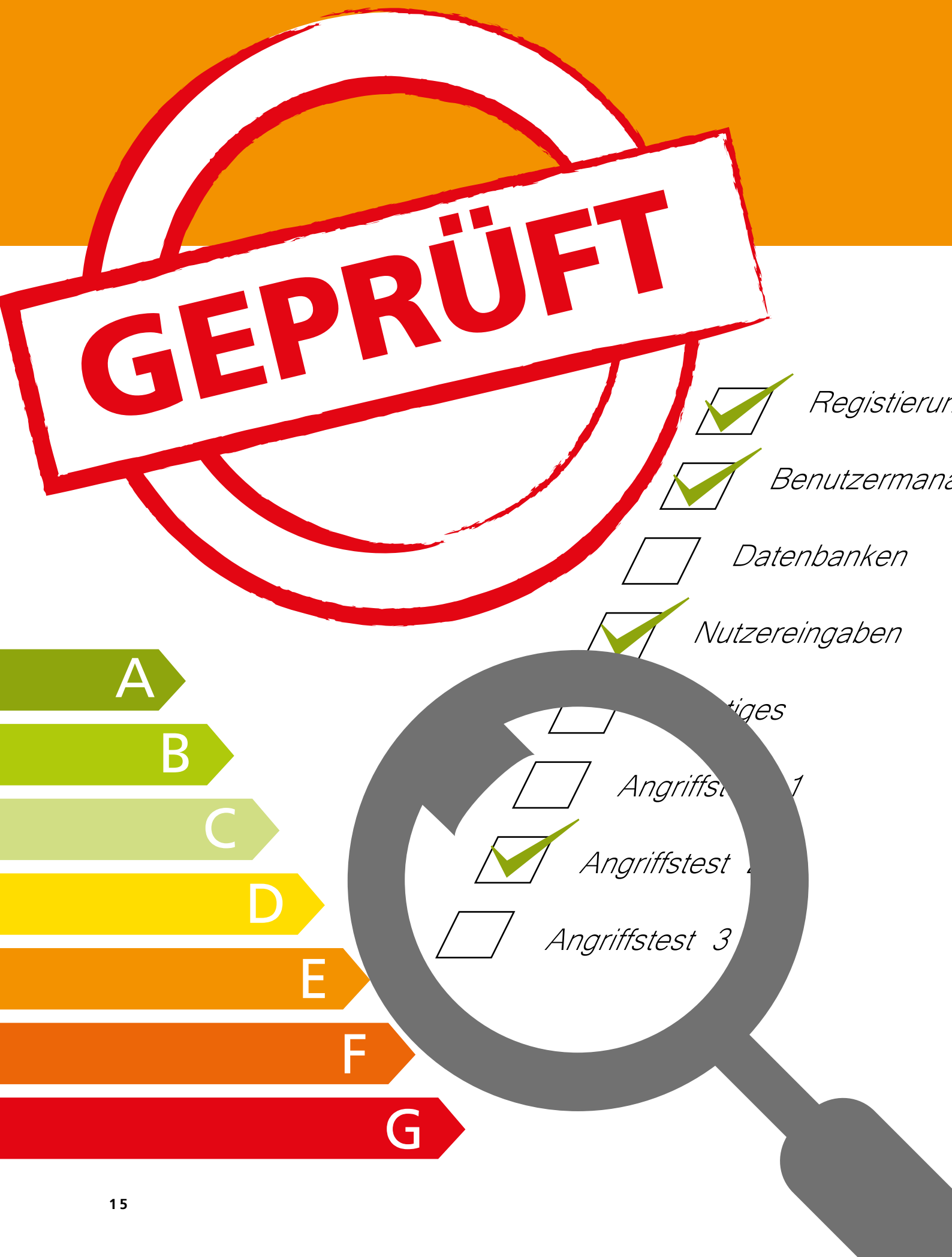
Lässt man den unrealistischen Anspruch der präzisen Messbarkeit von IT-Sicherheit fallen, lassen sich Anforderungen an die Verwirklichung von Indikator-Metriken aufstellen, die mittel- bis langfristig erreichbar scheinen.

Handlungsempfehlungen

Die angestrebten Messverfahren sollten zu annähernd gleichen Ergebnissen führen, auch wenn sie unterschiedliche Personen oder Unternehmen durchführen (intersubjektiv nachvollziehbar). Eventuell ist hierfür die Einführung von neuen, anerkannten Messeinheiten und Industriestandards erforderlich. Die Resultate sollten sich so aufbereiten lassen, dass sie unterschiedliche Perspektiven zulassen und von unterschiedlichen Stakeholdern praktisch nutzbar sind. Essenziell ist dabei eine entscheidergerechte Aufbereitung, damit das angestrebte bzw. das erreichte Sicherheitsniveau je Bedrohung schnell erfasst werden kann und die damit verbundenen Kosten quantifizierbar sind. Weitere Ergebnissichten könnten die der Wertschöpfungspartner und die der Kunden sein. Eine Priorisierung nach Perspektive (z. B. Anwendersicht) oder nach bekannten Risikoklassen / -umgebungen, z. B. vergleichbar mit Safety-Erwägungen in der Luftfahrt, könnten weitere hilfreiche Leitideen zur Kategorisierung sein. Wünschenswert für Messverfahren wäre zudem, dass diese geschäftsprozessabhängige Parameter berücksichtigen können. Außerdem sollten sie die Gesamtlösung betrachten und nicht nur die IT allein. Schlussendlich sollten Messverfahren auch die Fähigkeit berücksichtigen, Geschäftsaktivitäten bei einem Sicherheitsvorfall aufrecht zu erhalten (»security business continuity management«). In jedem Falle sollten die Messverfahren empirisch belegt oder zumindest in Zukunft empirisch belegbar oder falsifizierbar sein, um somit wissenschaftlichen Ansprüchen Genüge zu tragen und die Sinnhaftigkeit der abzuleitenden Entscheidungen gewährleisten zu können.

Als initiale Herangehensweise für Messverfahren diskutierten die Gesprächsteilnehmer die Erstellung von Modellen für Softwaresicherheit. Zu klären wäre, ob es sinnvoll ist, von den realen Prozessen auszugehen und daraus zunächst ein Gesamtmodell zu entwickeln, das die Grundlage für ein Messverfahren bildet. Alternativ könnte man auch mit der Betrachtung von Teilgrößen beginnen oder eine Messbarkeit ganz ohne Modell-Basis anstreben. Weitere Ansätze sind denkbar und noch zu diskutieren.

Ein weiterer diskutierter Vorschlag ist die Verpflichtung zu »Breach Notifications«. Mit diesen sollen Unternehmen staatliche Stellen über Sicherheitsvorfälle informieren, um verlässlichere Aussagen über die Anzahl, Schwere und Art dieser Vorfälle zu erhalten. Teilweise gibt es heute bereits inoffizielle Absprachen zwischen Unternehmen, die solche Notifications miteinander teilen. Eine Institutionalisierung hätte vor allem den Vorteil, dass größere Datenmengen erhoben werden könnten. Gleichzeitig muss jedoch auch ein verantwortungsvoller Umgang mit diesen sensitiven Daten gewährleistet werden.


GEPRÜFT
Registrierun
Benutzermana
Datenbanken
Nutzereingaben
tiges
Angriffst 1
Angriffstest
Angriffstest 3
A
B
C
D
E
F
G

2.3 LEICHTGEWICHTIGE ZERTIFIZIERUNG

Heutige Software-Zertifizierungsschemata wie Common Criteria gelten als wenig praxistauglich. Der Aufwand rentiert sich in der Regel nur für Hochsicherheitssoftware – in der Regel auf einem Absatzmarkt, für den die Zertifizierung verpflichtend eingefordert wird. Bei vielen Produkten, die mit einer Common Criteria-Zertifizierung auf den Markt kommen, sind außerdem nur kleine Teilkomponenten wirklich zertifiziert worden, teilweise mit Schutzzielen, die aus sehr wohlwollenden, schwachen Angreifermodellen abgeleitet wurden. Eine Beurteilung der Sicherheit des Gesamtprodukts ist damit unmöglich, die Zertifizierung erscheint hier oft als Werbeargument. Ganz ähnlich wie bei den Anforderungen zur Messbarkeit braucht es in diesem Bereich einen gänzlich neuen Ansatz: Dabei müssen Kriterien leicht anwendbar, aussagekräftig, intersubjektiv nachvollziehbar und änderungsfreundlich sein – bei plausiblen Kosten. Solch ein neues Zertifizierungsschema sollte auch den stark erweiterten Anforderungen der Lieferkettensicherheit gerecht werden.

Handlungsempfehlungen

Hohe Anforderungen für Software-Zertifizierungsverfahren zu formulieren ist letztlich Wunschdenken, wenn diese nicht geerdet werden. Die Vermutung ist plausibel, dass es sich angesichts der Komplexität und Heterogenität von Software auch hier um ein schwieriges Problem handelt – darauf weisen auch mannigfach gescheiterte Versuche hin, die Zertifizierung von Software zu reformieren bzw. neu zu gestalten.

Es wird deswegen empfohlen, zunächst eine gründliche Defizitanalyse zu starten, die mit genügend Zeit und Ressourcen ausgestattet sein sollte. Die Erforschung und Entwicklung einer schnell durchführbaren, aussagekräftigen und änderungsfreundlichen Zertifizierung von Software mit plausiblem Aufwand ist dann dringend geboten, eventuell mit Input aus unternehmensinternen Good Practices proprietärer Herkunft. Wenn das Zertifizierungsschema zudem kostengünstig wäre und Software-Lieferketten berücksichtigen würde, wäre dies ein großer Schritt für alle Stakeholder von den Anbietern über die Zertifizierungsdienstleister bis zu den Kunden.

Auch wenn bei einem neuen Zertifizierungsverfahren nichttechnische Elemente einbezogen werden – wie die Zertifizierung der Prozesse, unter denen Software erstellt wurde, oder gar die Zertifizierung der Akteure – muss weiterhin berücksichtigt werden, dass das Ziel der Zertifizierung auf den Gegenstand bezogen formuliert werden muss, also auf nachprüfbare Eigenschaften der Sicherheit von Softwarekomponenten und -produkten ggf. unter Berücksichtigung des jeweiligen Anwendungskontexts. Des Weiteren muss eine Zertifizierung bezogen auf einen oder mehrere Einsatzkontexte erfolgen. Eine Sicherheitszertifizierung ohne die Betrachtung eines solchen Kontexts ist naturgemäß nicht zielführend.


ANALYST 1,
3 JAHRE ERFAHRUNG
ANALYST 3,
8 JAHRE ERFAHRUNG
ANALYST 2,
1 JAHR ERFAHRUNG

2 HERAUSFORDERUNGEN UND EMPFEHLUNGEN

2.4 FLEXIBLE SICHERHEITSPROZESSE

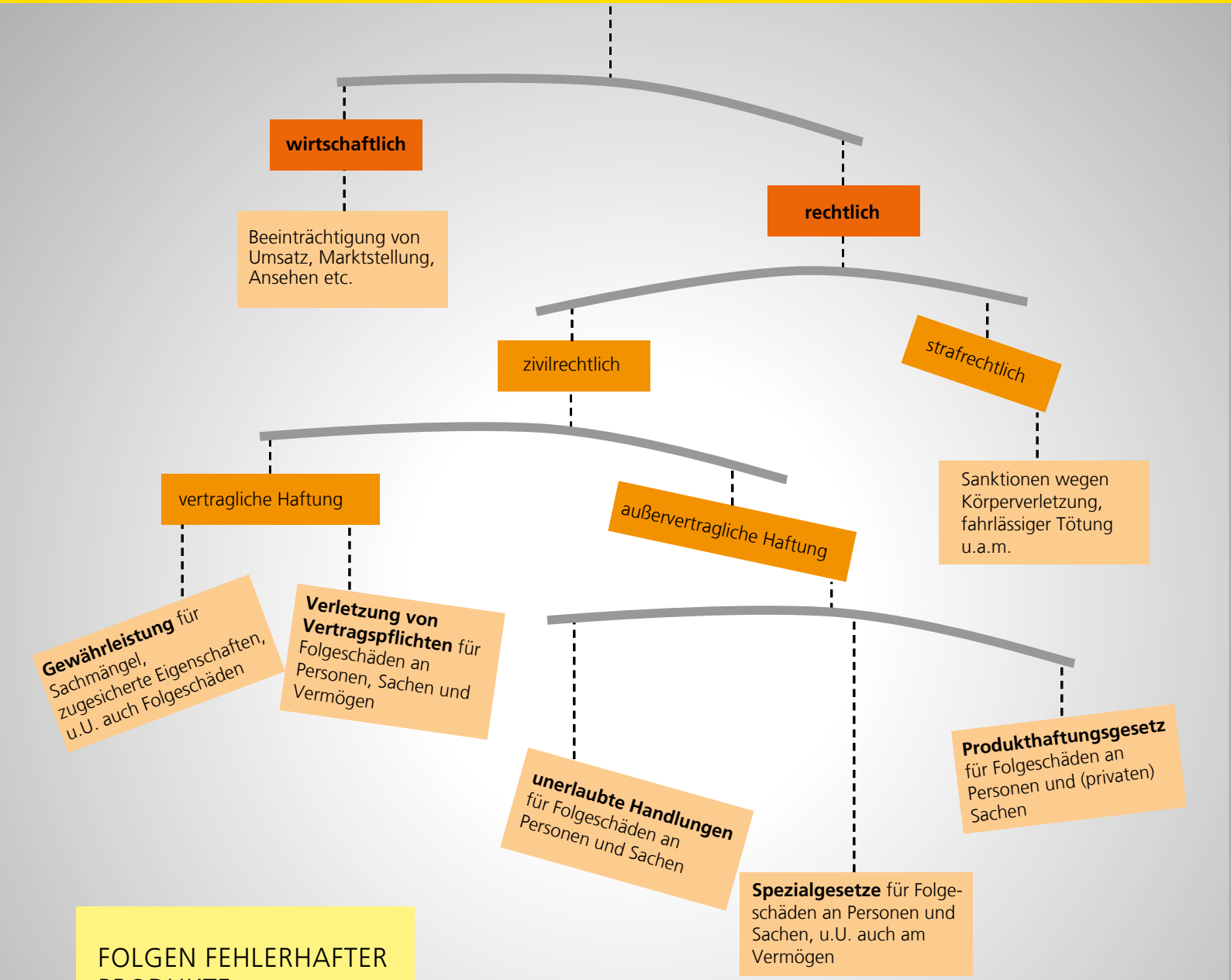
Während die meisten großen global agierenden Hersteller dem Thema Security bereits einen großen Stellenwert einräumen, verwendet ein Großteil der kleinen und mittleren Softwarehersteller immer noch keine klar definierten Prozesse zur Entwicklung sicherer Software. Zwei Drittel der Softwarehersteller folgen keinem definierten Sicherheits-Lebenszyklusprozess (InformationWeek 2012). Dies ist das Ergebnis einer Umfrage bei amerikanischen Softwareherstellern und gilt für deutsche Unternehmen wohl in derselben Größenordnung. Dr. Carsten Ochs hat im Rahmen des vom Bundesministerium für Bildung und Forschung geförderten Forschungszentrums »European Center for Security and Privacy by Design« (EC SPRIDE) eine ähnliche Umfrage gemacht, die diese Annahme bestätigt. Die Befragungsergebnisse weisen darauf hin, dass zwar große Softwarehersteller allesamt definierten Sicherheits-Lebenszyklusprozessen folgen, kleinere und mittlere Unternehmen (KMU) den Aufwand zur Etablierung eines solchen Prozesses jedoch offenbar scheuen. Für KMU ist die Rentabilität solcher Prozesse nicht offensichtlich und sie nehmen fälschlicherweise an, dass diese Prozesse nicht mit den neuen agilen Methoden der Softwareentwicklung vereinbar seien.

JEDER ANALYST HAT EINEN ANDEREN FOKUS UND FINDET ANDERE SCHWACHSTELLEN

Handlungsempfehlungen

Wie können sichere Softwareentwicklungsprozesse auch für kleine und mittelgroße Software-Produzenten attraktiv werden, deren Softwareentwicklungskultur sehr unterschiedlich ist? Die Gesprächsteilnehmer plädieren für eine Umkehrung der Anpassung, d.h. Verfahren für sichere Softwareentwicklungsprozesse richten sich zukünftig an Rahmenbedingungen der kleineren und mittleren Unternehmen aus. Insbesondere lassen sie sich verschlanken und sind zukünftig an unterschiedliche Vorgehensweisen und Rahmenbedingungen anpassbar. Konkret müssten die Sicherheits-Lebenszyklusprozesse unterschiedliche, skalierbare methodische Ansätze auf die einzelnen heterogenen Software-Design-Prozesse abbilden, Risikotransparenz herstellen, neue Architekturen und Entwicklungsumgebungen nahtlos unterstützen, Verantwortung expliziter verteilen, Schnittstellensicherheit unterstützen, Messpunkte für die Messbarkeit und Zertifizierung liefern, sichere Wiederverwendung ermöglichen, Methoden für Code von Drittanbietern wie Open Source vorweisen und in bestmöglicher Weise Analysewerkzeuge integrieren. Ebenso ist es essenziell, die Lieferkettensicherheit vollumfänglich zu berücksichtigen.

Zusammenfassend empfehlen die Teilnehmer, Hemmnisse für definierte Sicherheitsprozesse in der Software-Herstellung systematisch zu analysieren und abzubauen. Dies ist umso wichtiger für KMU, die diese bisher kaum einsetzen, was negative Auswirkungen auf die gesamte Lieferkettensicherheit hat.



FOLGEN FEHLERHAFTER PRODUKTE

Quelle: www.ndt.net/article/dgzfp/dach57/bauer/bauer.htm 1997

2 HERAUSFORDERUNGEN UND EMPFEHLUNGEN

2.5 BEANTWORTUNG DER HAFTUNGSFRAGE

Die Haftung für Fehler in Softwareprodukten gilt als eine der wichtigsten offenen Fragestellungen.[4] Eine Internet-Recherche bezüglich Haftung für Software bringt Formulierungen hervor wie »… Produkthaftung für medizinische Software ungeklärt …«, » … ungeklärt ist dagegen die Haftung für fehlerhafte Public-Domain-Software und Freeware …«, »…Übertragbarkeit auf Software ungeklärt …«, »…Richter sind ratlos …«. Dies ist nicht wirklich überraschend, denn seit der Industrialisierung ist das Recht bis auf wenige Ausnahmen ein Technikfolger, es hinkt natürlicherweise der technischen Entwicklung hinterher. Außerdem werden Gesetze, Richtlinien und Rechtsprechung bezüglich Datenschutz und Softwaresicherheit derzeit in verschiedenen Ländern teilweise sehr unterschiedlich gehandhabt.

Die Gesprächsteilnehmer sehen, dass eine Weiterentwicklung des Rechts viele Aspekte zu berücksichtigen hat: Einerseits kann es nicht im Interesse der Softwareindustrie sein, dass die klare Regulierung der Haftung zur Innovationsbremse wird und damit beispielsweise nationalen Marktteilnehmern jegliche weitere Marktchancen verwehrt. Andererseits hätte eine klare Haftungsregulierung das Potenzial, die Wettbewerbsfähigkeit zu steigern, denn bei klarer Regelung der Haftungsfrage wollen alle Kunden und Zwischenhändler ihr Risiko minimieren. Dadurch könnte die Attraktivität eines Produkts steigen, wenn für die Anwender im Schadensfall die Aussicht auf eine Entschädigung besteht. Viel wichtiger als die Zahlung von Entschädigungen selbst ist jedoch der Anreiz, den die bloße Verpflichtung zu ihrer Zahlung im Ernstfall setzt. Softwareentwicklungsunternehmen müssten solche Risiken kalkulieren und bilanzieren. Nötige Investitionen in Softwaresicherheit könnten somit direkt gegenfinanziert werden, da sie das Risiko mindern und somit die nötigen Rückstellungen senken.

Handlungsempfehlungen

Um den Rechtsrahmen für Haftung von Software weiterentwickeln zu können, erscheint ein anreizorientierter Diskurs unter Beteiligung von Rechtswissenschaft, Software-Industrie und Anwendern zielführend. Nachdem hier ein grundsätzlicher Konsens gefunden ist, könnten staatliche Stellen zu einem späteren Zeitpunkt in die Diskussion einbezogen werden und die parlamentarische Willensbildung vorbereiten. Von Beginn an sollten beim Diskurs Haftungsfragen für Privacy-Vorfälle einfließen, wie den unerwünschten Abfluss von Daten. Mögliche Ergebnisse reichen von einer garantierten Minimalsicherheit bis hin zur vollumfänglichen Haftung als Gerätesicherheit für Software, also ähnlichen Gesetzen wie bei der Safety-Regulierung, z. B. für Medizintechnik oder den Flugzeugbau. Zur Beförderung eines fairen Wettbewerbs gilt es außerdem zusammen mit anderen Ländern internationale Mindeststandards für Datenschutz und Softwaresicherheit zu schaffen.

Zusammenfassend regten die Workshop-Teilnehmer einen Diskurs über die Frage der Haftung für Softwarefehler an, an dessen Ende mehr Rechtssicherheit für alle Stakeholder stehen sollte. Eine Analyse der jahrzehntelangen Erfahrungen von Safety-orientierten Industrien wie dem Flugzeugbau und anderen regulierten Branchen könnte erste wichtige Denkanstöße liefern.

4 Michael Waidner, Michael Backes, Jörn Müller-Quade (Hrsg.): Entwicklung sicherer Software durch Security by Design; SIT Technical Report, Fraunhofer Verlag, München 2013; https://www.sit.fraunhofer.de/fileadmin/dokumente/studien_und_technical_reports/Trendbericht_Security_by_Design.pdf

Bedrohungsquelle

Angriffsvektoren

Schwachstellen

Sicherheits-
maßnahmen

Angriff

Angriff

Angriff

Schwachstelle

Schwachstelle

Schwachstelle

Schwachstelle

Maßnahme

Maßnahme

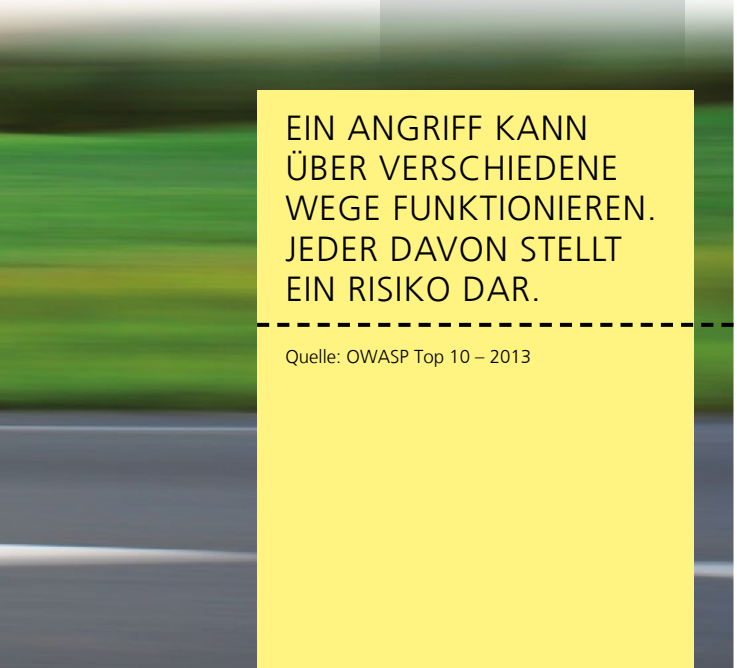
Maßnahme

2 HERAUSFORDERUNGEN UND EMPFEHLUNGEN

technische Auswirkungen

Auswirkungen auf das Unternehmen

Ressource	Auswirkung
Funktionalität	Auswirkung
Ressource	Auswirkung



EIN ANGRIFF KANN ÜBER VERSCHIEDENE WEGE FUNKTIONIEREN. JEDER DAVON STELLT EIN RISIKO DAR.

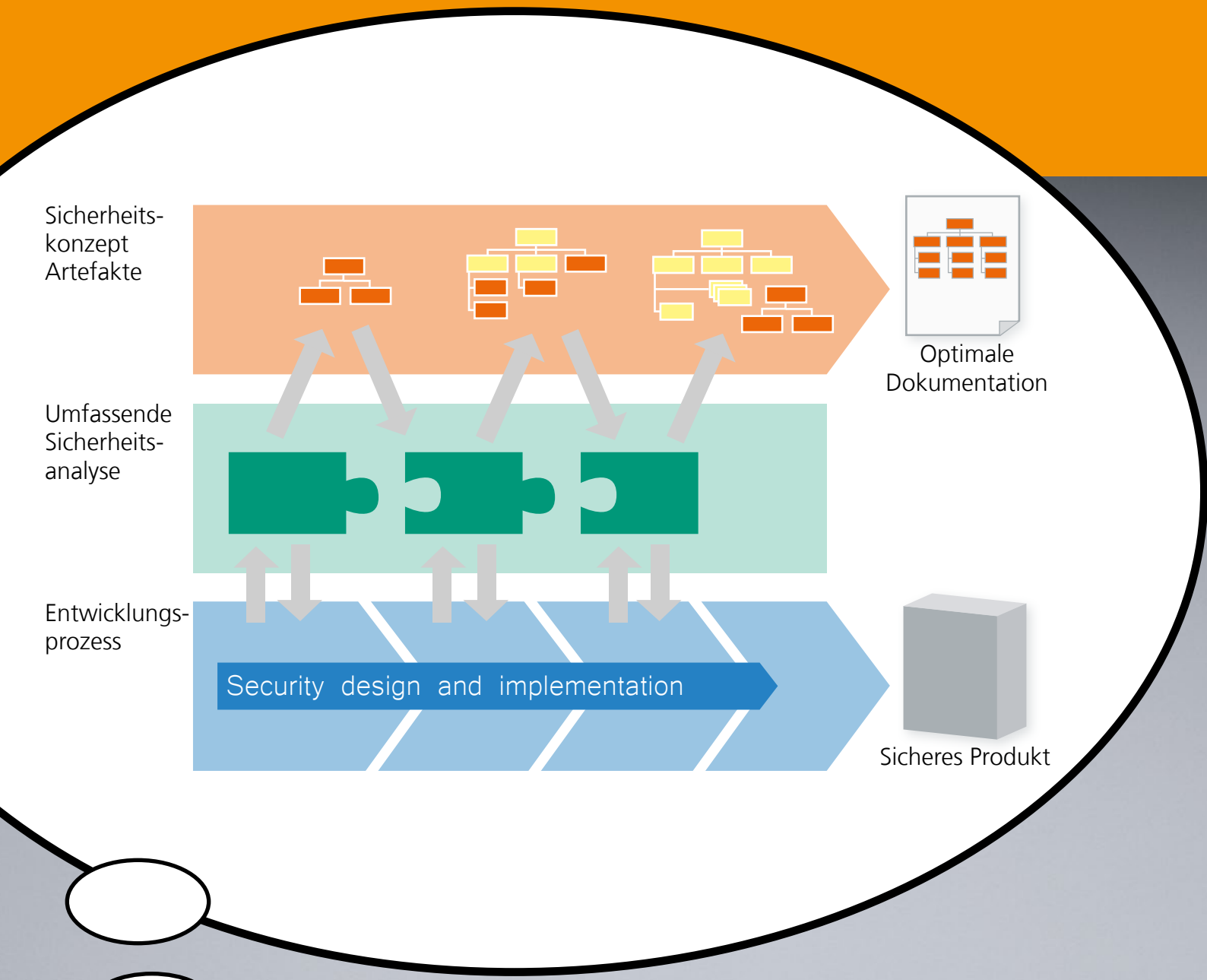
Quelle: OWASP Top 10 – 2013

2.6 MEHR SICHERHEIT FORDERN

Die Öffentlichkeit scheint sich daran gewöhnt zu haben, dass täglich IT-Sicherheitsvorfälle bekannt werden. Angesichts der zunehmenden Abhängigkeit unseres gesamten Wirtschaftssystems von funktionierender und sicherer Software überrascht es, dass dies scheinbar auch für Unternehmen als Softwarekunden gilt – warum wird Softwaresicherheit nicht verstärkt eingefordert? Ein zweites wirtschaftspolitisches Thema ist scheinbar komplett aus der öffentlichen und der Wahrnehmung der Fachwelt verschwunden: Die Versicherbarkeit von Software. Die Softwaresicherheit stellt selbst für Versicherer derzeit ein noch so schwer zu kalkulierendes Risiko dar, dass in diesem Bereich so gut wie keine Versicherungsprodukte existieren.

Handlungsempfehlungen

In den einschlägigen Vergaberichtlinien wie die »Ergänzenden Vertragsbedingungen für die Beschaffung von IT-Leistungen« (EVB-IT) und »Vergabeverordnung für die Bereiche Verteidigung und Sicherheit« (VSVgV) sollte ein Mindestniveau von IT-Sicherheit von Software festgeschrieben und in jedem Einzelfall plausible, nachprüfbare IT-Sicherheitsanforderungen von Software eingefordert werden. Das Mindestniveau an IT-Sicherheit in Softwareprodukten zeigt sich auch in den Reaktionsprozessen der Hersteller und Zulieferer auf Sicherheitsvorfälle wie neu aufgedeckte Lücken, laufende Angriffe oder sicherheitsrelevante Forschungsergebnisse. Eine nachweisbar schnelle und wirksame Reaktion ist bei Vergabeentscheidungen ein wichtiges Kriterium. Für kritische Bereiche sollte sie festgeschrieben werden.



VISION EINER SYSTEM-ÜBERGREIFENDEN METHODE DES SECURITY ENGINEERING

2 HERAUSFORDERUNGEN UND EMPFEHLUNGEN

Die Gesprächsteilnehmer fordern die Politiker auf, in der einschlägigen Gesetzgebung in Deutschland und in der EU Anreize für ein Mindestniveau von IT-Sicherheit von Software zu schaffen und es für kritische Bereiche festzuschreiben.

Generell müssen von der Politik finanzielle Anreize geschaffen werden, um die Erforschung von neuen Methoden und Mechanismen für sichere Software zu fördern. Das Thema »Versicherbarkeit von Software-Risiken« gehört ebenso dringlich auf die Forschungsagenda mit dem Ziel, auf Softwaresicherheitsrisiken zugeschnittene Versicherungsprodukte zu entwickeln, die beispielsweise auf vorhandenen Konzepten der Versicherbarkeit von IT-Risiken basieren. Ohne neue, effektive Metriken werden solche Produkte jedoch nicht möglich sein.

Zusammenfassend plädieren die Gesprächsteilnehmer für wirtschaftspolitische Anstrengungen zur Erhöhung der Softwaresicherheit, die die Nachfrage nach sicherer Software vor allem durch Vergaberichtlinien und monetäre Anreizsysteme anregt und das Problem der Versicherbarkeit von Softwaresicherheit nachhaltig löst.

2.7 AUSBILDUNG

Die Komplexität des Problemfeldes Softwaresicherheit verschärft sich. Dies ist die Hauptmotivation für eine große Anstrengung in Richtung Ausbildung.

Die Komplexität von Software nimmt durch »Code- und Systemexplosion« zu – diese stellen immer höhere Ansprüche an das Abstraktionsvermögen des Menschen. Kein Mensch kann alle Codezeilen einer Unternehmensanwendung, also mehrerer Millionen Zeilen Codes, allein erfassen und verstehen. Software gehört damit zu den komplexesten Entwicklungen der Menschheit.

Die Komplexität von Software nimmt in einer weiteren Dimension zu, nämlich durch die zunehmende Heterogenität und fortschreitende Zerfaserung der Einsatzgebiete, der Ausführungsumgebungen und -bedingungen, der Interaktion (z. B. machine-to-machine), der Angreiferstärke. Damit nehmen auch die Anforderungen an die Softwaresicherheit zu. Als eindrückliches Beispiel sei hier das Betriebssystem Linux genannt: Es kann von Privatpersonen für einfache Anwendungen wie die Textverarbeitung verwendet werden oder als gehärtetes Betriebssystem für Bankapplikationen. Im wachsenden App-Markt wird dies noch extremer: Schadcode-Apps nisten in trügerisch-sicheren Clouds, Bestandssoftware wird erweitert oder ersetzt oder in nicht dafür vorgesehenen Umgebungen eingesetzt. Open Source-Software, die keinen Support und kein Update mehr erfährt, dient als Basis oder essenzieller Baustein von unternehmenskritischer Software und für kritische Infrastrukturen unserer Gesellschaft.

Zu guter Letzt wird eine zunehmende Komplexität der Abhängigkeiten beobachtet. So durchdringen Cloud Computing, Apps, Cyber Physical Systems und mobile Endgeräte mehr und mehr Lebensbereiche wie Kommunikation, Arbeitswelt, Mobilität, Industrie, Freizeit, Gesundheitswesen, Energieversorgung. Die Grenzen zwischen diesen Welten verwischen, was eindrücklich neue Begriffe wie »BYOD – bring your own device« und »Car-Infotainment« aufzeigen.

DER TYPISCHE SOFTWAREENTWICKLER

36 Jahre

zwei bis drei Kinder

logisch, detail-verliebt, mäßig extrovertiert

Quelle: www.drdobbs.com/architecture-and-design/who-are-software-developers/240162014

Handlungsempfehlungen

Die Gesprächsteilnehmer schlagen die folgenden Maßnahmen zur Verbesserung und Verstärkung der Ausbildung vor: Die Tätigkeit und das berufliche Umfeld der (sicheren) Softwareentwicklung sollte für junge, technikinteressierte Menschen – vor allem auch Frauen – attraktiv gestaltet und entsprechend beworben werden. Sichere Software-Erstellung sollte Pflichtfach und Querschnittsthema in der Informatik-Ausbildung auf allen Ebenen werden. Training-on-the-job für einschlägig Berufstätige sollte eine feste Verankerung erhalten, beispielsweise durch regelmäßigen Austausch inklusive Besprechungen realer Vorfälle. IT-Sicherheitsweiterbildung muss als andauernder Prozess strategisch in den Unternehmen verankert werden. Die fortschreitende Komplexität erfordert lebenslanges Lernen – z. B. indem die »Programmierlizenz« abläuft, falls der Nachweis des regelmäßigen Besuchs von Workshops fehlt. Die Sicherheit komplexer Software kann nur von Teams sicher gestellt werden – die Teamkompetenzen aller Beteiligten, von den Programmierern bis zu den Managern, müssen stetig erweitert und stabilisiert werden. Neue Technologien erfordern oftmals ein komplettes Um- und Anlernen von neuen IT-Sicherheitsautomatismen und Vorgehensweisen, hierfür müssen Ressourcen eingeplant und zur Verfügung gestellt werden.

Die Gesprächsteilnehmer stellen zusammenfassend fest, dass die Komplexität von Software und deren Sicherheit nur noch in Teams von sehr gut ausgebildeten Personen bewältigt werden kann, Technik allein genügt definitiv nicht. Doch gerade solche Fachleute fehlen. Hier braucht es verstärkte Bemühungen in der Ausbildung auf mehreren Ebenen.

3 SCHLUSSBETRACHTUNG

Angesichts der großen Gefahr, die von Schwachstellen in Softwareprodukten ausgeht, ist Softwaresicherheit von großer Bedeutung und Dringlichkeit. Ob Cyberterrorismus, Computer-Kriminalität oder Wirtschaftsspionage – ohne verstärkte Anstrengungen von Politik, Forschung und Wirtschaft werden sich die aktuellen und kommenden Herausforderungen des digitalen Zeitalters nicht bewältigen lassen. Softwaresicherheit bildet deshalb einen wesentlichen Baustein, um Bürger, Unternehmen, die öffentliche Verwaltung und die Gesellschaft als Ganzes wirkungsvoll vor IT-Angreifern zu schützen.

Die entworfenen Handlungsmöglichkeiten stellen noch keine endgültige Roadmap dar, denn die einzelnen Maßnahmen müssen auch sinnvoll miteinander verbunden werden. Sie bilden jedoch einen vielversprechenden Ausgangspunkt für eine zielgerichtete Diskussion, die zeitnah zu konkreten Schritten und Erfolgen führen kann. Einige der skizzierten Punkte wie die Ausbildung oder die Klärung rechtlicher Fragestellungen bedürfen sicher längerer Zeiträume, andere Maßnahmen wie die Schärfung der Forschungspolitik und die Berücksichtigung von Softwaresicherheit im Rahmen von öffentlichen Ausschreibungen lassen sich schon mittel- bis kurzfristig umsetzen.

Gelingt es in Deutschland und Europa, die Anstrengungen zum Ausbau sicherer Softwareentwicklung zu bündeln, verspricht dies nicht nur die Sicherung des erreichten Wohlstands, sondern eröffnet auch unternehmerische Chancen sowie die Möglichkeit zur Schaffung wichtiger Standortvorteile.

Redaktion
Michael Waidner, Eric Bodden,
Michael Kreutzer, Sven Türpe,
Oliver Küch

Layout
Mona Bien

Anschrift der Redaktion
Fraunhofer SIT
Presse- und Öffentlichkeitsarbeit
Rheinstraße 75
64295 Darmstadt
Telefon 06151 869-282
Fax 06151 869-224

redaktion@sit.fraunhofer.de

Bildquellen
iStock: S.5-9, 17, 21-23
Fotolia: S.11-13, 21-22, 28
Getty images: S.25
Alle anderen Abbildung: