Toni Gansel
(toni.gansel@protonmail.com)
ist Testexperte für Application Security Testing. Seit mehr als einem Jahrzehnt unterstützt er dabei, qualitativ hochwertige Software zu entwickeln. Zurzeit ist er bei Sogeti Deutschland angestellt.

Sicherheit: Das Zugpferd der Softwarequalität

Die Welt der Informationstechnologie ist heute komplexer und umfangreicher, als sie es jemals war – Steigerungen sind absehbar. Mithilfe der IT verwalten wir alles Verwaltbare, steuern Routen, Einkäufe sowie Finanzmärkte. Sie ist tief in unsere Gesellschaft integriert und hat sie fundamental verändert. Die neuen Technologien wie Cloud-Computing, Internet of Things und Machine Learning sind die jüngsten Errungenschaften und ihnen haben wir Smarthomes, Industrie 4.0 und selbstfahrende Fahrzeuge zu verdanken. Die Durchbrüche bezüglich der Go und Starcraft II spielenden Computer sind kaum zu unterschätzen. Computer werden kreativ! Sie werden mehr Platz in unseren Leben einnehmen.

In einem Interview erläuterte Elon Musk kürzlich seine größte Befürchtung für das Unternehmen Tesla:

"I think one of the biggest concern for autonomous vehicles is somebody achieving a fleet-wide hack. In principles, if someone was able to say hack all the autonomous Teslas, they could say – I mean just as a prank – they could say 'send them all to Rhode Island' – across the United States … and that would be the end of Tesla and there would be a lot of angry people in Rhode Island." [Mus]

Die Welt der IT-Sicherheit ist heute viel umfangreicher und komplexer, als sie es jemals war – Steigerungen sind absehbar. In einer Welt, in der Daten die wichtigste Ressource sind, werden Daten gestohlen. Ob es private Bilder auf dem Smartphone, unveröffentlichte Serien von HBO oder das Wissen der NSA sind: Der Schaden ist beträchtlich und betrifft Privatpersonen genauso wie Unternehmen und Staaten. Es ist ein Problem der ganzen Gesellschaft.

Der Cyberspace wird heute nicht mehr von den stereotypischen IT-Nerds ohne Sozialleben beherrscht. Stärkere Interessengruppen wie Unternehmen und Staaten haben die Bühne betreten und einen sehr aktiven Part eingenommen. Finanzielle Mittel, technische Fähigkeiten, Motivationen und Rechtsempfinden sind undurchsichtige Eigenschaften im Cyberspace. Welche Daten auf welche Weise vor wem geschützt werden sollten, kann sehr unterschiedlich beantwortet werden, und die Interessen der Teilnehmer im Cyberspace konkurrieren. Es wäre naiv anzunehmen, dass man kein potenzielles Ziel sei und nichts Schützenswertes hat.

Staatliche Organe haben dies erkannt. Sie versuchen, Schutzmechanismen für den Cyberspace zu implementieren, und bauen sogar Cyberarmeen auf. Die EU verpflichtet mittels Datenschutz-Grundverordnung (General Data Protection Regulation, [GDPR]) die Unternehmen dazu, sorgfältig abzuwägen, welche Daten erhoben werden, und diese auch zu schützen. Den Cyberspace sicher zu gestalten, ist eine Mammutaufgabe. Sie betrifft alle Teilnehmer und alle Teilnehmer haben ihren Beitrag zu leisten. Nicht weil sie altruistisch sind, sondern egoistisch.

Im World Quality Report 2017 [WQR17] wird auf eine Studie von Gartner [Gar14] verwiesen, die besagt, dass 84 Prozent aller Sicherheitslücken auf der Anwendungsebene zu finden sind. Im Jahr 2010 hat die OWASP Foundation (Open Web Application Security Project) das erste Mal eine Liste mit den Top 10 der kritischsten Sicherheitslücken in Webanwendungen erstellt [OWASP10], in diesem Jahr wird sie voraussichtlich aktualisiert [OWASP17]. Die aktuellen Sicherheitslücken und deren Reihenfolge

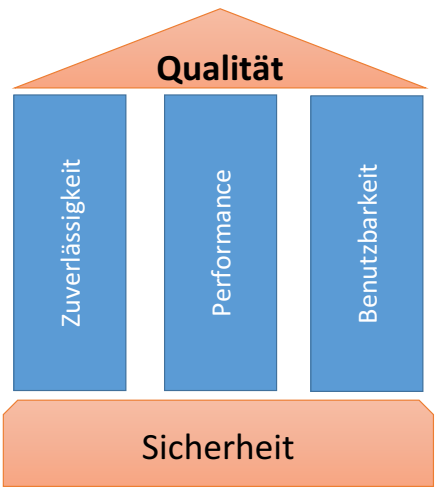

Abb. 1: Das Fundament der Qualität ist die Sicherheit

unterscheiden sich kaum von denen aus der ersten Version. Es ist bekannt, was unsere Anwendungen unsicher macht und wie man dagegen vorgehen kann. Nicht nur OWASP auch andere Institutionen wie das SANS-Institut (SysAdmin, Networking and Security) oder das Bundesamt für Sicherheit in der Informationstechnik (BSI) beobachten und dokumentieren die Gefahren im Cyberspace ausführlich.

In Softwareentwicklungsprojekten hat die IT-Sicherheit einen schwierigen Stand: Die Investitionen bringen keinen sichtbaren Mehrwert. Ganz im Gegenteil: Wenn man kostspielige Sicherheitstests durchführen lässt, ist die Beseitigung der identifizierten Sicherheitslücken oft der teurere Teil. Hier entstehen Kosten für einen Wert, der vielleicht während des gesamten Lebenszyklus der Software unsichtbar bleibt. Finanzielle Ressourcen sind begrenzt, neue Features werden erwartet und es gibt andere Aspekte, die dem Benutzer stärkeren unmittelbaren Wert liefern. Ob sich Investitionen lohnen und wie hoch sie sein müssen, ist nicht einfach zu entscheiden.

Dennoch war im vergangenen Jahr das Thema IT-Sicherheit die Top-Priorität in IT-Projekten und Unternehmen, dies wird sich wohl so schnell nicht ändern. Wer diese Investition unterlässt, riskiert einen komparativen Nachteil gegenüber den Mitbewerbern. Elon Musks Befürchtung ist keineswegs unbegründet.

Was häufig übersehen wird, ist, dass Sicherheitstests nicht nur vor Risiken schützen. Sie liefern auch einen unmittelbaren Mehrwert. Wie und wo dies geschieht, werde ich anhand einiger Beispiele erläutern, diese sind keineswegs erschöpfend, aber die Idee sollte sichtbar werden.

Risikoanalyse

Bevor mit Sicherheitstests begonnen wird, wird eine Risikoanalyse durchgeführt. In dieser Analyse werden die kritischen Prozesse und die schützenswerten Daten gewichtet. Bei dieser Gewichtung spielen Aspekte wie Projektziele, potenzieller Schaden durch Sicherheitsvorfälle und gesetzliche Verpflichtung eine Rolle. Es wird noch einmal verdeutlicht, wo der Fokus des IT-Projekts liegt, an welchen Stellen der Anwendung die Projektteilnehmer besonders sorgfältig sein müssen. Ebenso wird eine Grundlage für spätere Abuse-Cases und Negativ-Testfälle geschaffen.

Die neu gewonnenen Informationen helfen der Projektleitung dabei zu entscheiden, wann, wo und wie viel Geld investiert werden muss, um die Qualität und Sicherheit zu prüfen. Viele Unternehmen haben das Thema Risikoanalyse in ihrem Portfolio, aber auch unabhängige Institutionen wie das Bundesamt für Sicherheit in der Informationstechnik [BSI] oder OWASP [OWASPRRM] bieten hier Werkzeuge und Methoden an.

Denial of Service

Wenn man damit beginnt zu analysieren, wie es möglich ist, die Anwendung vollständig oder teilweise außer Betrieb zu nehmen (Denial of Service), wird zwangsläufig die Infrastruktur aus der Perspektive eines Angreifers betrachtet. Diese Perspektive legt Schwachstellen in der Anwendungsinfrastruktur und Performance-Engpässe offen. Sie betrachtet die Anwendungsinfrastruktur ganzheitlich und zeigt jede einzelne konfigurierbare Komponente auf. Durch diese Perspektive sieht man deutlich, wo welche Probleme entstehen können.

Konkrete Fragen, die aus dieser Betrachtung hervorgehen, sind beispielsweise: Ist der Message-Broker redundant? Wie reagiert die Anwendung, wenn der Datenbank-Verbindungspool erschöpft ist? Wie reagiert der Loadbalancer, wenn er mehr Anfragen erhält, als er beantworten kann? Was passiert, wenn die Festplatte des Servers voll ist? Wie reagieren Webservices auf Programmcode anstelle von Daten? Haben wir die optimale Caching-Strategie für unseren Applikationsserver-Cluster?

Die Erkenntnisse aus dieser Betrachtung helfen häufig dabei, die Infrastruktur performanter und sicherer zu gestalten. Ressourcenlöcher und Nadelöhre werden bei *Denial of Service*-Angriffen aufgespürt. Die Hersteller von Infrastrukturkomponenten bieten meistens Dokumentationen an, in denen erläutert wird, wie die Komponente sicher konfiguriert wird.

Statische Quellcodeanalyse

Um die Sicherheit einer Anwendung auf Quellcode-Ebene zu analysieren, empfiehlt sich die statische Quellcodeanalyse. Hierzu gibt es zahlreiche Tools, die man direkt in den Build-Prozess oder der Entwicklungsumgebung der Entwickler integrieren kann. Diese Tools identifizieren meistens wesentlich mehr als nur Sicherheitslücken. Mit ihnen kann man auch andere Programmierfehler unterschiedlichster Arten aufdecken und Programmierstandards setzen und prüfen.

Durch den Einsatz solcher Tools unterstützt man die Entwickler dabei, besser zu programmieren, die technischen Schulden gering zu halten und die Wartbarkeit des Quellcodes langfristig zu erhöhen. So wird Entwicklungszeit und damit Geld für zukünftige Änderungen gespart.

Welches Tool hierbei eingesetzt wird, sollte sorgfältig abgewogen werden. Wenn es nicht von den Entwicklern angenommen wird, ist es schwer, positive Effekte zu erzeugen. Viele bieten ein zentrales Dashboard und eine Integration in die jeweilige IDE, einige sind kostenlos, andere sehr teuer. Viele dieser Tools bieten die Möglichkeit, eigene Regeln zu formulieren. Wobei dieses Feature erfahrungsgemäß eher selten genutzt wird. Eine ausführliche Liste solcher Tools findet man auf Wikipedia [Wiki].

Eingabevalidierungen

Fehler, die *Injections* oder *Cross Site Scripting*-Angriffe zulassen, sind ein Indikator für eine fehlende oder fehlerhafte Eingabevalidierung. Fehler dieser Art führen die OWASP-Top-10 an und werden damit kritisch bewertet. Leider treten solche Fehler in diversen Ausprägungen immer wieder auf.

Dies ist nicht nur ein Problem für die Sicherheit, es hat auch negative Auswirkungen auf die Datenqualität. Wenn Daten ungeprüft gespeichert werden, wird meistens auch viel Datenmüll erzeugt. Darüber hinaus hat man hier die Möglichkeit, die Benutzerfreundlichkeit zu verbessern. Clientseitig kann man dem Benutzer schon früh sichtbar machen, welche Werte in welchem Format erwartet werden. Diese sollten im Anschluss aber zwingend serverseitig geprüft werden.

Das Thema mit der Eingabevalidierung ist leider nicht ganz einfach zu lösen, aber es gibt hier zahlreiche unterstützende Tools und Eigenschaften, die man auf unterschiedlichen Ebenen einsetzen kann.

Um die clientseitigen Schutzmechanismen von Webanwendungen zu umgehen, gibt es sehr raffinierte Tools wie *BurpSuite* oder *OWASP ZAP*. Die Entwicklerkonsole gängiger Webbrowser reicht allerdings vollkommen aus. Daher muss eine Eingabevalidierung serverseitig erfolgen, wobei nicht-gesetzte *Constraints* auf den Datenbankfeldern nicht nur aus Sicherheitsperspektive ein Armutszeugnis sind.

Eine Eingabevalidierung darf in keiner Anwendung fehlen, weil sonst die Benutzer frustriert werden und unsaubere Daten speichern. Die hierdurch erzeugte Sicherheit ist eine Gratis-Zugabe. Als weitere Maßnahme sollte man die sicheren http-Header setzen.

Sensitive Informationen

Leider erlebt man es noch häufig, dass Fehlermeldungen schlecht durchdacht oder implementiert sind. Im schlimmsten Fall sind sie nicht nur unverständlich, sondern legen auch systeminterne Informationen offen, häufig über *Stacktraces* oder *SQL-Statements*. In diesem Fall hat man eine grobe Sicherheitslücke.

Eine Fehlermeldung sollte die Benutzer darüber aufklären, was schief gelaufen und was zu tun ist. Dies sollte in einer freundlichen, verständlichen und konsistenten Sprache geschehen. Diese Anforderung an die Formulierungen erhöht nicht nur die Benutzerfreundlichkeit, sondern auch die Sicherheit.

Was bei dem Thema *Sensitive Data Exposure* ebenfalls gerne übersehen wird, ist, dass Entwicklerkommentare im HTML-Quelltext nichts zu suchen haben. Kommentare können an anderen Stellen verfasst werden und sollten niemals für die Endanwender sichtbar sein.

Um sensible Informationen nicht offenzulegen, sollte bei allen Projektmitgliedern ein entsprechendes Bewusstsein geschaffen werden. Es ist nicht immer bekannt, dass dies ein Sicherheitsrisiko darstellt.

Nachdem die Projektbeteiligten hierüber aufgeklärt worden sind, kann man diese Lücken gezielt identifizieren und beseitigen.

Fazit

Die Sicherheit einer Anwendung ist kein isolierter Aspekt. Sie ist eine Eigenschaft, die sich durch die ganze Anwendung zieht. Sie ist Bestandteil der Benutzeroberfläche, der Infrastruktur und Softwarearchitektur. Die Fähigkeiten der Programmierer und die Programmierstandards im Projekt sind Einflussfaktoren und zugleich Maßstab der Sicherheit.

Gute, solide und hochwertige Software ist immer auch sichere Software. Ein Mangel an Sicherheit ist ein Mangel an Qualität. ■

Literatur & Links

[BSI] BSI-Risikoanalyse, siehe:
https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzSchulung/WebkursIT-Grundschutz/Risikoanalyse/risikoanalyse_node.html
[Gar14] Gartner Studie (kostenpflichtig), siehe:
https://www.gartner.com/doc/2856020/maverick-research-stop-protecting-apps
[GDPR] Sogeti-Bericht über die General Data Protection Regulation, siehe:
https://www.sogeti.de/gdpr
[Mus] über Interview mit E. Musk bei der National Governor's Association im Juli 2017, zu Security ab Minute 59:20, https://www.youtube.com/watch?v=2C-A797y8dA
[OWASP10] OWASP-Top-10 2010, siehe: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2010
[OWASP17] OWASP-Top-10 2017, siehe:
https://owasp.blogspot.de/2017/08/owasp-top-10-2017-project-update.html
[OWASPRRM] OWASP-Risikoanalyse, siehe:
https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology
[Wiki] Werkzeuge zur statischen Codeanalyse, siehe:
https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis
[WQR17] World Quality Report, siehe:
https://www.sogeti.com/explore/press-releases/world-quality-report-2016-2017/