Kapitel 1
Einführung und Motivation

Zwei Großkonzerne erstellten im Jahre 2004 zusammen ein Kundenportal, mit dem Kunden ihre eigenen Stammdaten und die bestellten Services bequem verwalten konnten. Allerdings gestattete das Portal den Kunden auch die Verwaltung der Daten *anderer* Kunden – durch einfachste Manipulationen an den URLs der herunter geladenen Seiten. Gewieftere Angreifer konnten sich mit wenig Aufwand Administratorrechte verschaffen, und zwar im gesamten internen Netzwerk des Konzerns (Details findet man bei [CCC]). Am Ende musste das Portal für gut zwei Monate abgeschaltet und neu implementiert werden. Die Rede ist vom Online Business Solution Operation Center (OBSOC), dem Portal der Deutschen Telekom zur Verwaltung von Kundendaten. Wie kann es sein, dass wichtige, von großen Konzernen entwickelte Systeme so unsicher sind?

Im Juli 2005 werden von einem amerikanischen Verarbeiter von Visa- und MasterCard-Transaktionen 40 Millionen Kundendaten entwendet – teilweise sogar mit PIN-Nummer. Wie kann es sein, dass auf den Desktops der Mitarbeiter dieser Firma hochbrisante Kundendaten herumliegen und gestohlen werden können? Attacken durch Buffer Overflows, Viren und Trojanische Pferde bedrohen nach wie vor die meisten Maschinen und Benutzer weltweit und verursachen enorme Schäden. Wie kann es sein, dass heutige Betriebssysteme und ihre Besitzer so schnell an Eindringlinge verraten, statt ihnen Hilfestellungen beim Erkennen von Malware zu geben – oder noch besser: das Gefahrenpotential der Malware zu verringern? Warum führt die kleinste Lücke in Applikationen und Servern schon dazu, dass alles verloren ist?

In der Vergangenheit haben sich die Bemühungen der IT-Sicherheitsexperten im Zuge der Erfolgsstory des Internets vor allem auf die Netzwerksicherheit konzentriert. Zwar sind auch auf diesem Gebiet noch längst nicht alle Probleme gelöst – wir verfügen jedoch zumindest über eine Reihe von leistungsfähigen kryptografischen Modulen und Sicherheitsprotokollen, die, wenn richtig implementiert, eine zuverlässige Absicherung der Kommunikation zwischen zwei Rechnern bieten.

Die oben genannten Beispiele zeigen jedoch deutlich, dass nicht die Kommunikation zwischen Client und Server, sondern die *Software* auf Client- und Serverseite das Hauptproblem ist. Unzureichende Validierung der Eingaben der Clients führt dazu, dass böswillige Nutzer sich Rechte auf dem Server verschaffen können,

die ihnen nicht zustehen. Auf der anderen Seite sind die PCs legitimer Nutzer durch Malware gefährdet, die ihre persönlichen Daten ausspäht. Aber wie ist das möglich?

Wieso wird so viel unsichere Software entwickelt? Diese Frage wurde in der Vergangenheit schon in ganz verschiedener Weise beantwortet. Von der mangelnden Haftung der Softwarehersteller ist die Rede sowie vom Zeitdruck und unmöglichen Deadlines bei der Entwicklung. All dies spielt gewiss eine Rolle, aber es ist aus unserer Sicht nicht entscheidend. Im vorliegenden Buch vertreten wir die These, dass die folgenden Gründe für die Existenz der meisten unsicheren Software verantwortlich sind:

- An erster Stelle steht die mangelnde Wahrnehmung von Sicherheitsproblemen auf Seiten der Softwareentwickler. Der Mangel kann dabei auf ganz verschiedenen Ebenen existieren und reicht von Missverständnissen bezüglich der beteiligten Personen und Geschäftsvorgänge bis hin zu einem grundsätzlichen Missverständnis der Kommunikation in verteilten Systemen.
- Hinzu kommt mangelndes Wissen der Entwickler über Sicherheitstechniken und Sicherheitsarchitekturen, vor allem über das Zusammenspiel von Software, Infrastruktur und Benutzern. Verstärkt wird dieses Problem durch ein begriffliches Chaos auf Grund konkurrierender Standards und Techniken gerade im Sicherheitsbereich.
- An dritter Stelle steht ein Mangel an konkreten Mustern (Patterns) zur Lösung von Sicherheitsproblemen.
- An vierter Stelle – jedoch keineswegs weniger wichtig als die anderen – stehen die sicherheitsrelevanten Eigenschaften aktueller Betriebsysteme und Programmiersprachen, die meist gegen bekannte Prinzipien der sicheren Software-Entwicklung verstoßen (Fail Gracefully, Principle-of-least-Authority (POLA)).

Komplexe Frameworks, wie sie im Falle von OBSOC zum Einsatz kamen, beinhalten umfangreiche Hilfestellungen für die Erstellung von Webapplikationen. Sicherer Code wird hier zu einem guten Teil durch das Framework möglich gemacht. Nur müssen die Entwickler die Problemfelder zumindest erkennen und außerdem die Mechanismen des Frameworks verstehen und einsetzen können.

Der gegenwärtige Trend zu immer mehr Software im „embedded control"-Bereich mit Anwendungen zum Beispiel in der Haustechnik oder der Automobilindustrie wird das Sicherheitsproblem noch verschärfen. Was passiert, wenn die momentane Art der Softwareentwicklung auf diese Systeme übertragen wird? Welche Qualitäts- und Sicherheitsprobleme gefährden die Systeme und ihre Nutzer dann? In diesem Bereich tritt die Zuverlässigkeit als Teilaspekt sicherer Software stärker in den Vordergrund als bei Internet-Applikationen. Wo sind hier die Systeme, die Software verschiedenster Hersteller nebeneinander laufen lassen können, ohne dass sich die Komponenten durch die gemeinsame Benutzung von CPU, Speicher oder Geräten gegenseitig beeinflussen könnten? Virtuelles Memory vermag zwar die gröbsten Übergriffe zu verhindern, spielt jedoch innerhalb moderner Application Server und ihrer Virtual Machines eine immer geringere Rolle. Und die ausgeklügelten Virtualisierungstechniken von Mainframes, die auf

Hardware- und Softwareebene eine gute Isolierung garantieren, sind leider bei den kleinen Systemen nicht in dieser Art verfügbar.

In der letzten Zeit hat sich mit dem Spannungsfeld Usability versus Security ein weiterer wichtiger Aspekt sicherer Software bemerkbar gemacht. Angesichts der großen Zahl an semantischen Attacken wie Phishing oder Identity Spoofing, die die Benutzer momentan gefährden, zeigt sich, dass Software nicht mehr nur theoretische Sicherheit gewährleisten, sondern für den Nutzer auch verständlich und beherrschbar bleiben muss. Die Forderungen nach besserer Usability dürfen dabei nicht nur für Endbenutzer gelten, sondern müssen auf die sicherheitsrelevanten Werkzeuge der Entwickler selbst ausgedehnt werden.

Auf eine fundamentale Besserung der Sicherheit von Software zu hoffen, ist zumindest für die nächsten Jahre nicht angebracht. Die Rede von Bill Gates bei der RSA Konferenz 2005 ([Gates]) hat deutlich gemacht, dass sich die Maßnahmen von Microsoft zur Sicherung der Heimrechner auf den Aufbau einer verteilten Infrastruktur konzentrieren – sozusagen der Nachbau der Verwaltungsstrukturen, wie sie in großen Firmen heutzutage üblich sind. Große Data Center überwachen die einzelnen Rechner, erkennen Attacken und installieren Patches automatisch – gegen Gebühr.

Man darf aber nicht verkennen, dass dies eine Maßnahme nach bekannt gewordenen Attacken darstellt, also gegen so genannte Zero-Day Attacken nicht helfen wird. Wer sich nun von Open-Source-Software grundlegend Besserung erhofft, wird enttäuscht: Auch Linux unterscheidet sich hinsichtlich der Sicherheitsarchitektur keineswegs so deutlich von einem Windows basiertem System, dass hier Wunder zu erwarten wären. Die gleiche Beobachtung lässt sich im Vergleich der Open-Source Web-Browser Mozilla bzw. Firefox und dem Microsoft Internet Explorer machen.

Realistisch bleibt deshalb als Lösung auf lange Sicht nur eine auf die Bedürfnisse der Softwareentwickler zugeschnittene Ausbildung im Bereich sicherer Software. Diese Ausbildung muss:

- die Wahrnehmung von Sicherheitsproblemen massiv verbessern. Die Erfahrung hat gezeigt, dass dies am Besten durch häufige Sicherheitsanalysen geschieht, in Verbindung mit der Einführung von Sicherheitsprinzipien, die dann am konkreten Fall erprobt bzw. wieder entdeckt werden. Dieses Buch soll genau diesem Zweck dienen, nämlich Sicherheit als Gesamtsystem begreifbar zu machen und die Verbindung von Policies und Mechanismen zu erläutern.
- das Verständnis von Sicherheitsmechanismen und Techniken vermitteln. Die Erfahrung hat jedoch gezeigt, dass dies am Besten im Kontext eines realen Sicherheitsproblems geschieht und nicht davon losgelöst. Im Anschluss an eine konkrete Anwendung empfiehlt sich dann eine vertiefte Bearbeitung ausgewählter Fragestellungen. Typisches Beispiel ist hier der unterschiedliche Einsatzzweck von Kanal- bzw. Nachrichtenorientierter Kommunikation.
- sicherheitsrelevante Softwarearchitektur im Bereich Authentisierung und Authorisierung am konkreten Beispiel (z. B. Single-Sign-On im Portal) lehren.

- grundlegende Prinzipien sicherer Software wie POLA und deren technische Realisation auf den verschiedensten Ebenen (Betriebssystem, Sprache, Applikationsarchitektur) verstehen und in konkrete Softwaretechnik umsetzen helfen. Design Patterns für sichere Software helfen dabei.
- „weiche Faktoren“, wie die konzeptuellen Modelle von Benutzern, aber auch Entwicklern, als wichtig erkennen. Hier überschreitet man schnell die Grenzen der Informatik hin zu einer ganzheitlichen Betrachtung des Phänomens sichere Software. Dazu gehört beispielsweise die Kenntnis von Usability-Kriterien, aber auch die Prinzipien von Angriffen, die auf der Täuschung der Nutzer basieren, wie sie Kevin Mitnick in [Mit] eindrucksvoll dokumentiert hat.

Kernziel dieses Buches ist deshalb das Aufzeigen der Zusammenhänge zwischen Sicherheitsaspekten in der Infrastruktur, in der Software der Applikationen und Systeme sowie den Benutzern und Entwicklern dieser Systeme. Kryptografische Grundlagen und Protokolle sind für uns hauptsächlich interessant in Bezug auf die Konsequenzen, die ihre Eigenschaften für die Anwendungen nach sich ziehen.

Gleiches gilt für den Bereich der Netzwerksicherheit: Auch sie ist für uns nur interessant in Bezug auf das Zusammenspiel mit der Applikationssoftware, aber nicht als eigenständiges Gebiet. Zudem gibt es für diesen Bereich bereits etliche hervorragende Einführungen ([Bless], [Schä], [Schw]).

Die Schwierigkeiten bei der Wahrnehmung von Sicherheitsproblemen darf jedoch nicht darüber hinwegtäuschen, dass für die Erstellung sicherer Systeme eine gewisse Menge an sicherheitstechnischen Grundlagen nötig ist. Dieses Buch versucht diese Grundlagen auf eine Weise zu erklären, die es Software-Entwicklern erlaubt, sie auch in der Praxis zu erkennen und korrekt anzuwenden.

Der Aufbau des vorliegenden Buches gliedert sich wie folgt:

Zunächst wird in konkreten Fallstudien die Aufmerksamkeit für Sicherheitsprobleme geschärft und die Sicherheitsproblematik eingebettet in das Gesamtkonzept von Applikationen und Geschäftsvorgängen. Zum Beispiel untersuchen wir die Sicherheitsproblematik, die hinter großen Portalen steht und versuchen dabei zu zeigen, dass eine künstliche Einschränkung auf ein einziges Bedrohungsmodell (z. B. das Internet-Bedrohungsmodell) nicht mehr sinnvoll ist. Stattdessen muss der Gesamtkontext bestehend aus Benutzer, Nutzer-PC, Internet, Applikation, Entwickler und Infrastruktur betrachtet werden. Zur Gesamtsicht eines Sicherheitskonzeptes gehört auch seine Auswirkung auf die Betriebsorganisation (Ausbildung von Personal, Hilfe-Center bei Problemen mit der Sicherheit etc.), die meist erst sehr spät in den Blickwinkel von Entwicklern geraten und dadurch unvorhergesehene Kosten oder neue Sicherheitslücken verursachen können.

An die Fallstudien schließt sich ein Grundlagenteil an, der einzelne Aspekte aus den Fallstudien herausgreift und vertieft behandelt. Er beginnt mit einigen Anmerkungen zur Sicherheitsanalyse und ihren Komponenten, wie zum Beispiel Single-Sign-On oder Delegation. So wird die Problematik der Delegation von Requests innerhalb von Infrastrukturen erklärt sowie auf die Unterschiede von kanalbasierter und objektbasierter Sicherheit eingegangen. Die Sicherheitstechni-

ken verteilter Systeme und ihrer Middleware werden ebenfalls in diesem Grundlagenteil diskutiert. Darunter fällt etwa die Frage der Authentisierung in verteilten Systemen. Aus dem Bereich der Kryptographie werden wesentliche Bausteine zum Bau sicherer Protokolle vorgestellt, die Voraussetzung für das Verständnis der Sicherheits-Frameworks in späteren Kapiteln sind. Von besonderer Bedeutung ist dabei der Unterschied zwischen kanalbasierter Sicherheit (z. B. durch die Verwendung von SSL) und der objektbasierten Sicherheit durch die Verwendung von signierten Nachrichten. Das Verständnis objektbasierter Sicherheit ist die Voraussetzung für die Einführung von neuen, flexiblen Geschäftsmodellen durch Föderation.

Das Buch wird abgeschlossen mit drei Kapiteln, in denen wir wichtige Anwendungsfelder betrachten: Die Sicherheit von Content-orientierten Systemen, föderative Systeme und der Aufbau einer sicheren Infrastruktur für Internet-Applikationen. Rollenkonzepte, Benutzerberechtigungssysteme und die Problematik von Stellvertretung werden anhand von Content Management Systemen diskutiert. Als Sicherheitsmechanismus auf Content-Ebene wird der Einsatz von Label-basierter Security (LBAC) untersucht und rollenbasierter Sicherheit (RBAC) gegenübergestellt. Besonderes Interesse verdient LBAC nicht zuletzt deshalb, weil es eine wichtige Rolle innerhalb der User Access Control des neuen Vista Betriebssystems von Microsoft spielt. Es ergänzt hier die rein Permission-bezogene Zugriffskontrolle durch Acess Control Lists (ACLs).

Föderative Systeme beinhalten als wesentlichen Punkt die Übernahme einer bereits erfolgten Authentisierung durch andere. Wir werden sehen, dass sich dadurch nicht nur Ressourcen einsparen lassen, sondern durchaus auch eine Steigerung der Sicherheit erreichbar ist. Außerdem wird die Technik der web-basierten Föderation von Identität unter Wahrung von Privatheit gezeigt.

Im Bereich der Infrastrukur wird ein Konzept zur konsequenten Reduktion von Angriffsoberflächen innerhalb der DMZ sowie des Intranets vorgestellt. Aus der Architektur der DMZ werden Anforderungen für eine damit kompatible Applikationsarchitektur abgeleitet: Was kann/soll ein Application Server tun bezüglich Authentisierung? Nützt ein Proxy in der DMZ und welche Auswirkungen hat er für die Applikationsarchitektur? Wie weit ist die Applikationsarchitektur durch Regeln der DMZ betroffen? Die Reverse Proxy Architektur zur Authentisierung und Authorisierung wird an einem konkreten Produkt erläutert.

Dem aufmerksamen Leser ist wahrscheinlich nicht entgangen, dass die Aufzählung der Themen in diesem Band nur einen Teil der oben von uns erwähnten Problematik bei der Entwicklung sicherer Systeme umfasst. Ein zweiter Band wird sich unter dem Titel „Sichere Systeme“ mit den existierenden Software-Frameworks zur Absicherung von Systemen beschäftigen. Die so genannte End-to-End Security innerhalb von Firmen wird darin ebenso behandelt wie die Absicherung von Servern auf den jeweiligen Plattformen. Anforderungen aus der DMZ Architektur werden in Form von Sicherheitsframeworks in J2EE bzw. Java erneut aufgegriffen und gelöst. Neben diesen softwaretechnischen Grundlagen erweitert der zweite Band das Thema der Software-Sicherheit um zwei wesentliche Dimensionen: Usa-

bility und Security einerseits, und andererseits die Einschränkung von Autorität als Grundprinzip bei der Entwicklung sicherer Systeme.

Der vorliegende Band basiert in Teilen auf Lehrveranstaltungen der Autoren zum Thema Internet-Sicherheit an der Hochschule der Medien in Stuttgart sowie auf eigenen Erfahrungen bei der Entwicklung von Portalen und anderen Applikationen. Es richtet sich vor allem an Softwareentwickler und Studierende der Informatik. Aber auch Spezialisten der Netzwerksicherheit, die eine Brücke zur Software-Security schlagen möchten (was in der letzten Zeit vor allem im Bereich Input Validierung geschieht), werden hoffentlich davon profitieren. Der Folgeband „Sichere Systeme“ geht darauf aufbauend sehr viel tiefer auf die Konstruktionsprinzipien von Software und ihren Auswirkungen für die Sicherheit von Systemen ein und richtet sich in noch stärkerem Maße an die Entwickler von Software. Nun aber zunächst viel Spaß beim Studium der Grundlagen der Internet-Security aus Software-Sicht.