Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Walter Kriha · Roland Schmitz

Sichere Systeme

Konzepte, Architekturen und Frameworks

Springer

Prof. Dr. Walter Kriha
Hochschule der Medien
Nobelstraße 10
70569 Stuttgart
kriha@hdm-stuttgart.de

Prof. Dr. Roland Schmitz
Hochschule der Medien
Nobelstraße 10
70569 Stuttgart
schmitz@hdm-stuttgart.de

ISBN 978-3-540-78958-1 e-ISBN 978-3-540-78959-8

DOI 10.1007/978-3-540-78959-8

Xpert.press ISSN 1439-5428

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

© 2009 Springer-Verlag Berlin Heidelberg

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Einbandgestaltung: KünkelLopka, Heidelberg

Gedruckt auf säurefreiem Papier

9 8 7 6 5 4 3 2 1

springer.de

Inhaltsverzeichnis

Kapitel 1
Einführung und Motivation

Warum wird so viel unsichere Software entwickelt? In dem Buch „Internet-Security aus Software-Sicht – Grundlagen“ [KS] haben wir vor allem die Mängel in der Wahrnehmung von Sicherheitsproblemen betont und versucht, diese Wahrnehmung anhand von Fallbeispielen und Sicherheitsanalysen zu stärken. Gleichzeitig wurden dort die sicherheitstechnischen Grundbausteine, Protokolle und Dienste in einer solchen Form vorgestellt, dass sie für praktisch tätige Entwickler, Architekten und Infrastrukturspezialisten verständlich und nutzbar sind.

Mit der verbesserten Wahrnehmung von Sicherheitsproblemen und Grundkenntnissen in Kryptografie allein ist es jedoch nicht getan. Wie sichert man eigentlich konkret eine Server-Applikation für das Internet ab? Gemeint sind hier nicht mehr Infrastrukturmaßnahmen, welche die Angriffsfläche innerhalb der DMZ verkleinern, sondern die Serversoftware selbst und die Plattform, auf der sie läuft. Welche Strategien und Mechanismen müssen wir einsetzen, damit unsere Software die Businessmodelle erfüllt, sodass nur autorisierte Personen Zugriff erhalten? Das Wissen über die Strategien ist in den Köpfen erfahrener Systemarchitekten enthalten, oder in den Vorschriften zur Softwareentwicklung mancher Konzerne und nicht zuletzt in einzelnen wissenschaftlichen Papieren z. B. über den Bau sicherer Web-Server. Eine Aufgabe dieses Buches ist es, dieses Wissen zusammenzutragen, zu diskutieren, die jeweiligen Grenzen der Strategien aufzuzeigen und somit für die Entwickler nutzbar zu machen.

Heute entsteht Software meist auf der Basis von Frameworks, die bereits Sicherheitstechniken z. B. für die Autorisierung von Zugriffen eingebaut haben und deren Verständnis essenziell für den Bau sicherer Systeme ist. Dies betrifft Applikationsentwickler wie Systemingenieure. Die letzteren verwenden ebenfalls Frameworks, um Applikationen mit der jeweiligen Firmeninfrastruktur zu verbinden. Somit entstehen diffizile Abhängigkeiten zwischen Infrastruktur und Applikationen. Dieser Band hat daher als einen weiteren Schwerpunkt, die Absicherung von Enterprise-Software durch Frameworks wie z. B. EJB, JAAS etc. zu erklären.

Das Buch bleibt aber nicht bei der Betrachtung einzelner Teile von Sicherheitstechniken stehen, sondern versucht das Gesamtsystem zu betrachten. Ein Gesamt-

W. Kriha, R. Schmitz, *Sichere Systeme*.
DOI: 10.1007/978-3-540-78959-8_1, © Springer 2009

system besteht aus der Kombination von Betriebssystem, Programmiersprache, Applikationsarchitektur und nicht zuletzt den beteiligten Personen. Zwischen diesen Komponenten bestehen weitere Abhängigkeiten, und die Sicherheit der einzelnen Komponenten garantiert noch längst nicht ein sicheres Gesamtsystem. Leider sorgt jedoch umgekehrt eine einzige unsichere Komponente meist für die Unsicherheit des Gesamtsystems.

Der Einbezug der beteiligten Personen in das Sicherheitskonzept des Gesamtsystems mag manche überraschen, vor allem wenn deutlich wird, dass dies auch die beteiligten Entwickler, Administratoren etc. umfasst. Oft wird unter psychologischen Aspekten der Sicherheit lediglich auf die potenziellen Angreifer geschaut, und beim Anteil der Usability an einem sicheren System lediglich auf den Benutzer. Dabei haben Unverständnis, Druck sowie schlechte Werkzeuge auf Seiten der Entwickler und Administratoren ebenfalls einen entscheidenden Anteil an unsicheren Systemen. Nicht zu vergessen alte Programmiermythen wie die Mär vom Ausgleich technischer Defizite in Programmiersprachen durch „Disziplin". Wir müssen begreifen, dass gewisse Defizite immer wieder zu Sicherheitsproblemen führen werden. Täglich bestätigen z. B. neue Buffer-Overflow-Attacken diese Erkenntnis.

Sehr interessant sind in diesem Zusammenhang auch die Untersuchungen zur Risikowahrnehmung und ihrem Einfluss auf die IT-Sicherheit, wie sie von Volker Scheidemann [Schei] durchgeführt wurden. Die ungleich bessere Akzeptanz von Virenscannern im Vergleich zu Verschlüsselungsprodukten kann demnach nicht nur auf Probleme der Usability bei letzteren zurückgeführt werden. Es scheint grundsätzliche, psychologisch motivierte Defizite bei der Risikoeinschätzung zu geben. Wir werden im Kapitel zu den Attacken sowie zu Usability und Security daher versuchen, den Ansatz von Scheidemann im Bereich der Softwareentwicklung anzuwenden. Sollte sich auch hier der Einfluss von falscher Risikowahrnehmung zeigen, wäre ein weiterer Grund für unsichere Software (bzw. deren lange Lebensdauer) gefunden und der Stärkung der Wahrnehmung innerhalb der Usability müsste noch mehr Gewicht eingeräumt werden.

Aber die verkürzte Sichtweise auf ein Gesamtkonzept von Sicherheit hat natürlich eine alte Tradition, die erst mühsam überwunden werden muss. So erweitert z. B. das Konzept der „multi-lateralen" bzw. „mehrseitigen" Sicherheit nach Ranneberg et al. [Rann] die Sicherheit einer Geschäftslösung, indem es nicht nur nach der Sicherheit der anbietenden Firma fragt, sondern auch nach der Sicherheit des Konsumenten. Dadurch entstehen ganz neue Forderungen z. B. nach dem Schutz der Privatsphäre oder dem Recht auf Anonymität.

Bei den bisher angesprochenen Problemen befinden wir uns im „klassischen" Bereich der IT-Security, verstanden als Einsatz von Sicherheitstechnik, um die Einhaltung von Businessmodellen auf der Basis von Authentisierung, Autorisierung und Nachweisbarkeit zu gewährleisten, angereichert mit dem Wissen um die Problematik der Benutzbarkeit und die Wissensdefizite bei den Beteiligten. Das vorliegende Buch versucht jedoch noch einen Schritt weiter zu gehen, indem es einige unbequeme Fragen stellt:

Wieso versagt Software im Zusammenhang mit Sicherheitsproblemen häufig vollständig? Wieso scheint es bei Softwaresystemen keine Schadensbegrenzung zu geben, wie wir sie aus anderen Bereichen wie z. B. dem Automobilbau kennen? Dort bewirkt der Ausfall einer Komponente meist nicht mehr den völligen Verlust der Sicherheit. Mit diesen Fragen verlassen wir die typische Angriffsperspektive, aus der Sicherheit in erster Linie als Schutz vor Angriffen auf die Businessmodelle interpretiert wird. Nun geht es auch um grundsätzlichere Fragen der Zuverlässigkeit und Berechenbarkeit von Software unter dem Gesichtspunkt der „Safety", also dem sicheren Funktionieren von Software. Eine zentrale These dieses Buches ist, dass es letztlich Defizite der „Safety" – also grundlegende Mängel in der Art, wie wir unsere Software konstruieren – sind, die dann auch gravierende Sicherheitslöcher nach sich ziehen. Es sind also nach Ansicht der Autoren nicht nur Denkfehler bei der Verwendung von Security-Protokollen, die Sicherheitsprobleme verursachen. Diese gibt es zweifellos und wir werden versuchen, sie nach Kräften zu erklären und zu beseitigen. Aber dieser andere Bereich – hier „Safety" genannt, wird mit der völligen Computerisierung unserer Lebenswelt noch wesentlich wichtiger. Hier stößt die bloße Wahrnehmung von Angriffsmöglichkeiten und ihrer Abwehrmaßnahmen als Werkzeug zur Entwicklung sicherer Software an ihre Grenze und ein Konzept von Schadensbegrenzung wird nötig. Damit verbunden sind auch Konzepte der Schadensabschätzung bzw. der Beweisbarkeit von Schadensgrenzen.

Abbildung 1.1 versucht, das Verhältnis der Dimensionen Security und Safety innerhalb von Softwaresystemen abzubilden:

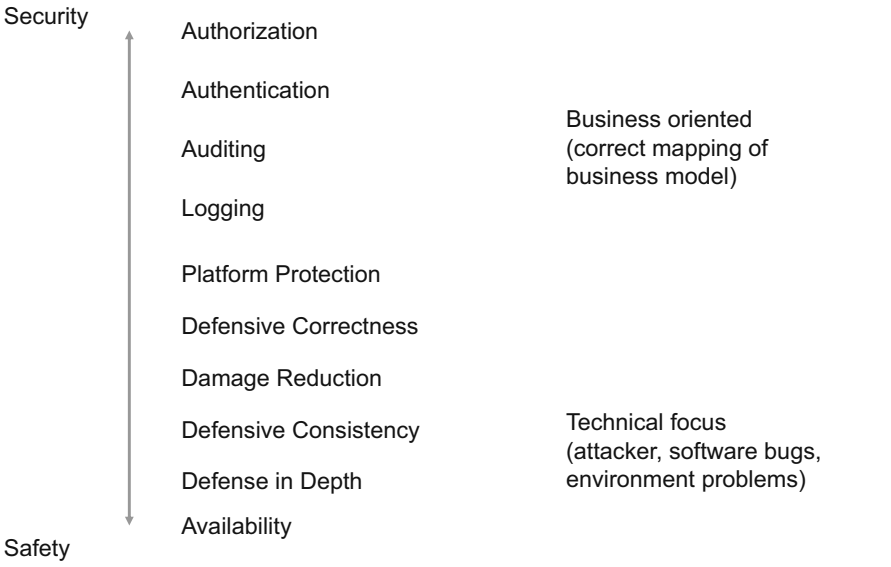

Abb. 1.1 Sichere Systeme als Kombination der Aspekte Security und Safety

Deshalb arbeitet dieser Band anhand der diskutierten Software-Frameworks die jeweiligen Mechanismen der Zugriffskontrolle (Zugriff hier verstanden als kausale Möglichkeit, Effekte zu erzielen) sowie deren Defizite heraus. Wir sehen uns also die Konzepte und Techniken im Einsatz an und untersuchen anschließend ihre Eigenschaften. Als Leitprinzip werden wir an jeder Stelle, an der ein Zugriff kontrolliert wird, die Frage nach der tatsächlich vorhandenen Autorität stellen (existieren zu viele Rechte?). Und wir werden die Zugriffskontrolle mit der Intention der Beteiligten vergleichen: Bewirkt die Kontrolle tatsächlich die Sicherheit, die gemeint ist?

Warum scheinen bestimmte Sicherheitsmängel innerhalb von Applikationen oder Systemen immer wiederzukehren? Zur Beantwortung dieser Frage untersucht das Buch bestimmte Fehlertypen und führt sie auf architektonische Ursachen in Systemen, Sprachen oder Applikationen zurück. Zu einer ganzheitlichen Betrachtung von sicheren Systemen gehört aber zwangsläufig auch die Betonung von „Sicherheit, die funktioniert" – ein Begriff, den Simson Garfinkel in seiner Studie zum Zusammenspiel von Usability und Security ([Garf]) geprägt hat: Die Anwendung von Sicherheitsprinzipien und ihre Implementation in realen Systemen mit realen Benutzern in realen Situationen. Wir werden im Laufe des Buches des Öfteren Sicherheitsmechanismen begegnen, die in der „wirklichen Welt" einfach nicht funktionieren. Sei es, weil die User-Interfaces unverständlich oder mühsam zu bedienen sind, sei es, weil Implementationen beim Einschalten des „sicheren Modus" leider unerträglich langsam werden. Aus diesem Grund laufen .NET-Server meist „fully trusted" und bei Java Enterprise Edition (JAVA EE) basierten Servern wird ebenfalls meist empfohlen, die Code-Based-Security auszuschalten. Oder sei es, weil die Modellierung der Konfiguration von Sicherheitskriterien so schlecht ist, dass selbst die Entwickler Fehler machen.

Nun zum Aufbau des Buches: Wir beginnen mit einer Untersuchung gängiger Attacken auf Plattformen und Applikationen sowie der Abwehrmaßnahmen. Zentrale Aussagen dieses Teils sind die Notwendigkeit formaler Beschreibung von Input und Output von Applikationen sowie die Reduktion der Applikationen und Modulen zur Verfügung stehenden Autorität. Außerdem wird die semantische Attacke als zukünftig dominierende Form dargestellt. Gedanken zur Sicherheit von Web2.0-Applikationen beenden das Kapitel zu Attacken. Am Beispiel von Web Application Firewalls untersuchen wir, was beim Auftreten von Sicherheitslücken in Legacy-Applikationen schnell getan werden kann. Ziel der Darstellung der Attacken ist jedoch in erster Linie, auf Grundsatzfragen hinzuweisen, z. B. warum eine Schwäche oder Lücke eigentlich so fatal für die Sicherheit des Gesamtsystems ist. Dazu werden sowohl lokale Angriffe als auch solche über das Netzwerk vorgestellt.

Im Anschluss an die Behandlung von Angriffen werden die Grundkonzepte, die hinter sicherer Software stehen, anhand einer Diskussion der klassischen Thesen von Salzer und Schroeder [SaSch] eingeführt. Wir gehen bei unserer Diskussion davon aus, dass die Plattformsicherheit und mit ihr das sogenannte Host-Bedrohungsmodell in Zukunft stark an Bedeutung gewinnen wird. Zwei entscheidende Begriffe werden in diesem Kapitel eingeführt: Das „Principle Of Least

Authority (POLA) als zentrale Maßnahme der Schadensbegrenzung und das Gegenteil davon: „Ambient Authority“ (s. z. B. [Wiki]) als frei schwebende, jederzeit verfügbare Möglichkeit, Schaden anzurichten. Alle bekannten Konzepte und Techniken zur Absicherung von Zugriffen (Access Control Lists, Sandboxes, Isolation durch Modi etc.) werden wir daraufhin untersuchen, inwieweit sie POLA beachten bzw. wieviel Ambient Authority zugelassen wird. Dabei werden wir das Prinzip der „Autorität“ im Sinne einer kausalen Möglichkeit, einen Effekt zu erzielen, vom Prinzip des „Rechts“ abgrenzen, hier verstanden als grundsätzliche Erlaubnis, etwas zu bewirken, was jedoch nicht automatisch die Möglichkeit dazu einschließt.

In einem speziellen Kapitel wird anschließend die Sicherheit von Plattformen (Betriebssystemen) und ihre Technik als Voraussetzung für die Sicherheit von Softwaresystemen behandelt. Wie sicher sind unsere Plattformen? Welche Mittel der Zugriffskontrolle verwenden sie? Welche Hardware-/Softwaremechanismen bewirken Sicherheit?

Neben Betriebssystemen stellen Sprachen einen wichtigen Teil der Sicherheit einer Software dar. Im Rahmen der Plattformsicherheit wird das Singularity-Betriebssystem von Microsoft kurz vorgestellt, da es eine rein auf Software basierende Isolationsstrategie verfolgt. Die Mächtigkeit von kontrollierten Referenzen auf Objekte zeigt sich daran, dass keine Hardware zur Virtualisierung nötig ist und auf grobe Isolationskonzepte wie Modi verzichtet werden kann. Die richtige Form von Softwareinstallation wird in diesem Zusammenhang ebenfalls vorgestellt.

Die Absicherung von Server-Applikationen wie Web-Servern etc. ist nach wie vor ein großes Problem. Das entsprechende Kapitel zeigt gängige Isolationstechniken sowie die Einengung des Zugriffs auf Backend-Systeme und die Problematik mächtiger, funktionaler Zugriffsidentitäten (Accounts). Im Kapitel zu Application-Servern wird auf diese Erkenntnisse aufgebaut.

Im Anschluss an die Plattformsicherheit werden sicherheitsrelevante Aspekte von Programmiersprachen unter spezieller Berücksichtigung von Java untersucht. Auf die gängigen Frameworks wie JAAS wird dabei natürlich eingegangen, ebenso auf die Code Access Security, wie sie durch die Java 2 Security gegeben ist. Eine wesentliche Aussage dieses Kapitels ist, dass die Code Access Security keinesfalls so unabhängig von der Architektur der Applikation ist, wie es zunächst bei Ansicht der externen Konfigurationsmöglichkeiten aussieht. Es wird gezeigt, dass hier bei Beginn der Applikationsarchitektur verschiedene Sicherheitsebenen eingeführt werden müssen, um zu verhindern, dass Applikationen über Systeminterna Bescheid wissen müssen bzw. zu viel Autorität erhalten. Der Stack-Walk-Mechanismus wird ebenfalls vorgestellt und bewertet. Interessanterweise existieren neben der ACL-basierten Zugriffsmechanik in Java durchaus weitere Sicherheitsmechanismen wie Namespaces und Object-Capabilities. Diese spielen in später behandelten Frameworks dann eine zentrale Rolle.

Anschließend an die sprachbasierte Sicherheit tritt das Thema Enterprise-Security ganz in den Vordergrund. Konzepte und Techniken zur Absicherung ganzer Infrastrukturen und Applikationen werden vorgestellt und kritisch hinterfragt. Das Verhältnis von Infrastruktur und Applikation in Bezug auf Sicherheit

wird untersucht und am Beispiel von JAVA-EE-Mechanismen nachvollzogen. Entscheidende Erkenntnisse sind dabei, dass Sicherheitscode zwar erfolgreich aus den Applikationen verbannt werden konnte, dass dadurch aber die Rolle der Infrastruktur extrem wichtig wird und Anpassungen der Infrastruktur nun ähnlich kritisch zu sehen sind, wie Änderungen an den Applikationen, etwa aus Gründen der Reorganisation. Zudem stellt die Delegation von Requests über mehrere Stufen einer Infrastruktur hinweg nach wie vor ein großes Problem dar.

Das Problem der Delegation wird anhand der sogenannten Proxy-Zertifikate aus dem Grid-Computing-Bereich erneut aufgegriffen. Dabei diskutieren wir die Frauge, ob durch vermehrten Einsatz von objektbasierten Sicherheitsmechanismen (Token, Zertifikate) die Infrastruktur wieder flexibler gestaltet werden könnte, indem eine semantische Trennung zwischen Business und Infrastruktur eingeführt wird.

Ein weiteres Ergebnis des Kapitels zur Enterprise-Security ist die Erkenntnis, welche zentrale Rolle der Applikation-Server im Gesamtkonzept sicherer Applikationen spielt. Dieser wird im Anschluss an die Enterprise-Security behandelt. Wir diskutieren unter anderem die wichtigen Interfaces zwischen Application-Server und Firmeninfrastruktur. Der Trend bezüglich sicherer Delegation von Aufrufen geht dabei klar in Richtung von tokenbasierten Mechanismen, mit denen Erzeuger und Transporteure von Requests klar unterschieden und authentisiert werden können. Auch auf spezielle Probleme der Application-Server wird dabei eingegangen, z. B. im Zusammenhang mit dem „Härten“ von Application-Servern.

Die Sicherheitsproblematik von Embedded-Control-Plattformen wird in Zukunft kaum zu überschätzen sein – getrieben durch die Ausbreitung der Rechner in sämtliche Lebenswelten. Der Zwang zur günstigen Produktion steht dabei oft in einem Spannungsverhältnis zur Isolation fremder Softwarekomponenten aus Sicherheitsgründen. Dies ist z. B. dann ein Thema, wenn die Anzahl der Steuergeräte in Fahrzeugen reduziert werden soll. In einem Kapitel zu Multi-Vendor-Komponentensystemen werden am Beispiel von OSGi und FINREAD zwei Architekturen zur Bereitstellung sicherer Services vorgestellt. Die dazu nötigen Isolationstechniken auf Basis virtueller Maschinen sowie getrennte Namespaces durch Class-Loader werden untersucht und bewertet. Das Konzept der signaturbasierten Softwaredistribution wird ebenfalls vorgestellt und kritisch hinterfragt.

Neben der klassischen End-to-End-Infrastruktur moderner Firmenapplikationen, die vom Browser des Kunden über die Midrange-Server bis zum Mainframe reicht, beinhaltet dieses Buch auch die Grundlagen föderativer Sicherheit auf Basis von Web-Services sowie die sicherheitstechnischen Grundlagen virtueller Organisationen am Beispiel von Grid Computing.

Web-Services dienen uns als Beispiele objektbasierter Sicherheit auf der Basis von signierten Nachrichten (Messages), und sie zeigen einen Weg aus der Abhängigkeit von der Infrastruktur. Da die Sicherheitsinfrastrukturen verschiedener Firmen ebenfalls sehr verschieden sein können, kann Sicherheit nicht mehr als reines Vertrauensverhältnis innerhalb einer Infrastruktur implementiert werden. Sicherheit muss daher auf höheren Ebenen – eben in den Nachrichten und Dokumenten – angesiedelt werden. In diesem Kontext wird auch die Notwendigkeit von

Sicherheitsabstraktionen für Applikationsentwickler gezeigt, die erst die Techniken objektbasierter Sicherheit einsetzbar machen. Grids sind von diesem Standpunkt aus lediglich als extreme Form von Föderation zu sehen. Das Konzept der virtuellen Organisation mit ihren Sicherheitsproblemen wird kurz dargestellt. Einige besonders mächtige Mechanismen aus der Grid-Security werden ebenfalls vorgestellt. Hier ist die zentrale These, dass Grids aufgrund ihrer extremen Sicherheitsproblematiken (wohl nur noch übertroffen von Peer-to-Peer-Netzwerken) zur Entwicklung neuartiger Sicherheitstechniken geradezu gezwungen sind.

Dieser Band behandelt eine Vielzahl von aktuellen Technologien im Zusammenhang mit sicherer Software. Im Vordergrund steht dabei sicher die Vermittlung von Kenntnissen zu aktuellen Frameworks und Techniken. Dennoch wird dabei auch versucht, die in den Plattformen, Sprachen und Frameworks eingesetzten grundlegenden Sicherheitsmechanismen und ihre Funktionsweisen herauszuarbeiten. Dabei stellt sich heraus, dass es nur eine kleine Zahl grundsätzlicher Techniken ist, auf denen die Sicherheit zumeist basiert, nämlich:

- Isolierte Räume und Vertrauenszonen,
- Access-Control-Listen und Referenzmonitore,
- Call-Verfolgung,
- Multi-Level-Security (Labelling, Tagging, Tainting),
- Namespace-basierte Isolation,
- Object Capabilities,
- kryptografische Methoden,
- modusbasierte Security und
- authentisierte Sessions.

Interessanter als die bloßen Mechanismen sind die Dimensionen, in denen sie sich unterscheiden: So funktionieren einige unabhängig von der Softwarearchitektur, während andere fester Bestandteil der Architektur werden. Entsprechend lässt sich die Zugriffskontrolle bei einigen dynamisch und schnell ändern, während andere Änderungen an der Software bedingen. Einige Mechanismen stellen Rechte permanent zur Verfügung, andere lassen Rechte in Abhängigkeit vom Bedarf fließen. Einige basieren auf Erlaubnissen in Form von Regeln, andere arbeiten kausal auf der Basis von Zugriffsmöglichkeiten über Referenzen oder signierten Vollmachten. Einige hängen an den beteiligten Daten, während andere auf der Codebasis operieren. Schließlich funktionieren einige eher lokal innerhalb von Vertrauenszonen, während andere in verteilten Systemen mit sich gegenseitig misstrauenden Partnern eingesetzt werden können.

Im letzten Teil des Buchs steht die theoretische Analyse der Sicherheitsprobleme von Softwaresystemen im Blickpunkt. In einem Grundlagenkapitel zur Konstruktion sicherer Software wird eine sehr wichtige These vertreten: Sichere Software benötigt Softwarebausteine, die nachweisbar und korrekt bestimmte Sicherheitskonzepte implementieren und dadurch eine Reduktion von Autorität vornehmen. Sichere Software ist daher nicht nur eine Frage von korrekten Sicherheits-Policies in Form von ACLs, sondern eine Frage der Softwarearchitektur auf unterschiedlichsten Ebenen der Granularität (einzelne Statements, Module,

Architekturmuster). Sicherheit ist demnach kein „Aspekt“, der beliebig externalisiert werden kann, sondern eine Kerneigenschaft, die in der Architektur der Software angesiedelt ist. In diesem Teil des Buches kommt Sicherheit sozusagen „nach Hause“ zur Software und wird als Softwareproblem erkannt – jenseits aller Kryptografie oder Protokolle. Man kann diese Erkenntnis noch radikaler formulieren: Unsichere Software ist ein generelles Qualitätsproblem der Softwareentwicklung und wird nicht nur durch schwierig zu verstehende Kryptografie oder fehlerhaft implementierte Sicherheitsprotokolle verursacht.

Anschließend wird eine konkrete Sicherheitsanalyse – hier von Mozilla/Firefox – vorgestellt. Innerhalb dieser Analyse werden die gemeldeten Sicherheitsprobleme untersucht, kategorisiert und anschließend auf grundlegende Probleme der Plattform, der Sprache oder der Applikationsarchitektur zurückgeführt. Am Ende steht die Erkenntnis, dass die Sicherheit komplexer Applikationen wie z. B. Browsern momentan außer Kontrolle ist und nur durch grundsätzliche Änderungen der Architektur sowie der Plattformsicherheit verbessert werden kann. Das Kapitel hat zudem den didaktischen Zweck, die bisher aufgebaute Wahrnehmung von Sicherheitsproblemen und deren Ursachen zu testen und bietet darüber hinaus reichlich Hypothesen für empirische Untersuchungen von Sicherheitsfehlern.

Daran anschließend wird ein alternatives Browsermodell basierend auf dem Konzept von Object-Capabilities vorgestellt, der sogenannte DARPA-Browser. Dieser Browser wurde explizit unter Berücksichtigung von Least-Authority-Prinzipien und einer sicheren Sprache entwickelt und dient als Forschungsinstrument für die Möglichkeiten und Grenzen solcher Architekturen. Zentrale Erkenntnis ist hier, dass es tatsächlich möglich ist, eine Architektur zu entwickeln, bei der bösartige Komponenten nur minimalen Schaden anrichten können. Außerdem wird klar, dass es innerhalb von Applikationen weitere Bereiche der Isolation geben muss.

Aus den obigen Analysen ergeben sich fast automatisch auch Fragen nach der Usability: Ist es wirklich so, dass beide in einem fast unversöhnlichen Gegensatz stehen, dass also mehr Security zwangsläufig zu einer schlechteren Bedienbarkeit führt? Welche Rolle spielt die System- bzw. Applikationsarchitektur dabei? Gibt es eine Alternative zu der momentan üblichen Masse an Sicherheitsdialogen mit unverständlichen Meldungen? Die zentrale Aussage dieses Kapitels ist die Behauptung, dass ohne eine massive Reduktion von Autorität kein wirklicher Gewinn an Sicherheit durch Verbesserungen der Benutzbarkeit möglich ist.

Der dritte Teil schließt mit der Betrachtung neuerer formaler Ansätze in der Sicherheitsanalyse sowie bei der Entwicklung von Sicherheitskomponenten als Softwarebausteinen. Das Konzept der Erreichbarkeit als Basis für Sicherheitsanalyse und -design wird anhand von Capabilities und erweiterten Take/Grant-Systemen vertieft. Wir reflektieren hier die Ansätze von Miller und Shapiro (s. [MS]) sowie Spiessens (s. [Spie07]) und ihren Versuch, durch Object-Capabilities eine Schadensbegrenzung innerhalb der Software einzuführen. Zentrale These dieses Kapitels ist die Behauptung, dass ohne die Entwicklung von Softwarekomponenten/-bausteinen zur Reduktion von Autorität weder die Entwicklung sicherer Software noch deren Analyse möglich ist. Diese Bausteine verwirklichen prüfbare Pattern

für Kollaboration (s. [Spie07]). Das Kapitel stellt außerdem Ansätze zur Modellierung von Sicherheitsaspekten in Infrastrukturen vor. Am Beispiel des „confused deputy“ stellt Fred Spiessens die Sprache Scoll und den Ansatz der automatischen Überprüfung von Sicherheitsmodellen (modell checking) vor.

Dieser Band richtet sich verstärkt an Softwareentwickler, z. B. in Enterprise-Umgebungen. IT-Security-Spezialisten, die traditionell eher im Bereich der Netzwerksicherheit zu Hause sind, erfahren durch dieses Buch eine Vielzahl von sicherheitsrelevanten Fehlerquellen und Ursachen im Bereich der Softwareentwicklung und können daher ihre Abwehrmaßnahmen besser planen. Aber auch der mehr theoretisch interessierte Leser wird im Bereich der capabilitybasierten Zugriffskontrolle und ihrer Modellierung hoffentlich wichtige Anregungen finden. In seiner fundamentalen Ausrichtung schuldet der Band sehr viel der sicheren Programmiersprache „E“ [Erights] sowie den Diskussionen um die Verwendung von Capabilities als Mittel der Schadensreduktion auf der Cap-talk-Mailing-Liste [Cap].

Wie für den ersten Band finden sich aktuelle Arbeiten, Vorlesungsunterlagen und Diskussionen sowie Fehlerkorrekturen auf http://www.kriha.de/krihaorg/sicheresysteme.html.

Literatur

[Cap] Cap-talk mailing list, http://www.eros-os.org/pipermail/cap-talk/

[Erights] Erights.org, Homepage der sicheren Sprache „E“

[Garf] S. Garfinkel, Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable, Dissertation, Massachusetts Institute of Technology (MIT) 2005, available online at http://www.simson.net/thesis/

[KS] W. Kriha, R. Schmitz, Internet-Security aus Software-Sicht – Grundlagen, Springer-Verlag 2008

[MS] M.S. Miller, J.S. Shapiro, Paradigm Regained: Abstraction Mechanisms for Access Control, Proceedings of 8th Eighth Asian Computing Science Conference (ASIAN’03), edited by Vijay Saraswat, Springer-Verlag 2003 (available online at http://www.erights.org/talks/asian03/)

[Rann] K. Rannenberg, A.Pfitzmann, G.Müller, Sicherheit – insbesondere mehrseitige IT-Sicherheit, http://www.wiiw.de/publikationen/Sicherheitinsbesonderemehrsei.pdf

[SaSch] J.H. Saltzer, M.D. Schroeder, The Protection of Information in Computer Systems, available online at http://www.cs.virginia.edu/~evans/cs551/saltzer/

[Scheid] V. Scheidemann, It won’t happen to me – Aspekte der Risikowahrnehmung und ihr Einfluss auf die IT-Sicherheit, in: Innovationsmotor IT-Sicherheit, Tagungsband zum 10. Deutschen IT-Sicherheitskongress, BSI 2007

[Spie07] F. Spiessens, Patterns of Safe Collaboration, Ph.D. dissertation, Department of Computing Science and Engineering, Université Catholique de Louvain, Feb. 2007, http://www.info.ucl.ac.be/~pvr/fsp_thesis.pdf

[Wiki] Ambient Authority, Wikipedia http://en.wikipedia.org/wiki/Ambient_authority

Kapitel 2
Angriffe

Glücklicherweise gibt es seit einiger Zeit ausgezeichnete Hilfen gegen das komplexe Problem der web-basierten Attacken. So behandelt z. B. Sverre Huseby [Hus] das Thema Input-Validierung für Web-Applikationen ausführlich. Auch der Guide von OWASP.org zur Entwicklung sicherer Web-Applikationen deckt die möglichen Angriffe sehr gut ab. Deshalb wollen wir hier nur einige wesentliche Angriffsformen behandeln, im Übrigen aber mehr Wert auf die Einordnung der Attacken in ein generelles Sicherheitskonzept legen. Außerdem wollen wir einige theoretische Überlegungen zur grundsätzlichen Bekämpfung dieser Attacken anstellen.

Durch das Phänomen des „Ubiquitous Computing", der vollständigen Durchdringung des Alltags und Lebensraums durch Computer, bekommt neuerdings auch die lokale Sicherheit der Geräte eine immer größere Bedeutung. So ist die Installation neuer Software zwar heutzutage meist über einen Netzwerkanschluss durchzuführen, aber die Frage, wie sich die Installation auf die lokale Sicherheit des Hosts[1] auswirkt, hat zwar nichts mehr mit „Web Security" im engeren Sinne zu tun, ist aber für unsere Diskussion dennoch von Interesse, da es sich bei diesen Hosts um Kundenrechner, also Teile des Gesamtsystems, handelt. Wir wollen daher auch lokale Angriffe besprechen.

Erreichen wollen wir durch diese Diskussion zum einen natürlich eine weitere Sensibilisierung für Web-basierte Angriffsmöglichkeiten. Zum anderen möchten wir einen theoretischen Rahmen für die Gefährlichkeit der Attacken aufspannen und schließlich auch die möglichen Abwehrmaßnahmen aufzeigen, die vom Bereich Netzwerk bis hin zur Softwarearchitektur, je nach Angriffsform, ganz unterschiedliche Formen annehmen können.

[1] „Host" wird synonym zu „Rechner" und „Node" verwendet.

W. Kriha, R. Schmitz, *Sichere Systeme*.
DOI: 10.1007/978-3-540-78959-8_2, © Springer 2009

2.1 Eine Übersicht der Attacken

2.1.1 Wie lassen sich die Angriffe klassifizieren?

Eine erste Möglichkeit zur Klassifikation der Angriffe ist die „Schicht“ oder „Ebene“, in der sie auftreten: Wir unterscheiden Angriffe auf das Netzwerk, Angriffe, die auf fehlende oder fehlerhafte Input-Validierung durch einen Web-Server abzielen oder aber konzeptionelle Angriffe, die die Kommunikationssession zwischen Client und Server angreifen.

Die Angriffe auf der Netzwerkebene sind häufig auf die Verfügbarkeit der Server-Applikation gegenüber dem Internet ausgerichtet, können aber auch die Verfügbarkeit des internen Netzwerks betreffen. Die Möglichkeiten, diese Angriffe durch reine Softwaretechniken abzuwehren, sind eher begrenzt. So sind Sites wie grc.com oder unlängst heise.de erfolgreich mit dem Effekt angegriffen worden, dass sie längere Zeit nicht zur Verfügung standen. Die Abwehrmaßnahmen beinhalten die Kontaktaufnahme zum ISP mit dem Ziel, die Netzwerkstränge, aus denen die Angriffe erfolgen, bereits vor dem Erreichen des eigenen Netzwerks zu blockieren. Dies hat zur Folge, dass legale User aus diesen Bereichen den Server ebenfalls nicht erreichen können. In diesem Fall hilft es einem ausgeschlossenen Kunden weiter, wenn er einen anonymisierenden Proxy verwendet, der sich in einem anderen IP-Adresssegment befindet.

Die weitaus größte Vielfalt an Angriffsmöglichkeiten nutzt hingegen Schwächen in der Validierung von Input. Darunter fallen Buffer-Overflow-Attacken, aber auch client- und serverseitige Cross-Site-Scripting-Attacken. Die Struktur dieser Angriffe ist eigentlich sehr einfach, ihre Vermeidung setzt jedoch ein klares Konzept in Softwaretechnik und Organisation der Entwicklung voraus. Diese Angriffsform kann extern über das Netzwerk, aber auch lokal erfolgen.

Die dritte Gruppe von Angriffen zeichnet sich dadurch aus, dass keinerlei netzwerktechnische oder softwaretechnische Fehler vorliegen. Das Gesamtsystem funktioniert wie erwartet, die Attacke läuft hingegen auf konzeptioneller Ebene: Der Benutzer irrt sich in Bezug auf den Adressaten oder den Inhalt einer Aktion. Programmierer haben hier serverseitig die Aufgabe, diese Attacken so weit es geht zu erschweren oder über eine Gesamtlösung auszuschließen. So lassen sich theoretisch Phishing-Attacken durch Signierung jeder einzelnen Transaktion unterbinden.

Eine andere Klassifizierung teilt die Angriffe in solche ein, deren Abwehr lediglich ein erweitertes Internetbedrohungsmodell voraussetzt, bei dem die Kommunikationskanäle, die verwendeten Protokolle und die Benutzer betrachtet werden und solche, bei denen ein hostbasiertes Bedrohungsmodell zum Tragen kommt (also zum Beispiel Angriffe durch kompromittierte Programme). Diese Unterscheidung ähnelt stark derjenigen zwischen „Netzattacke“ und „lokaler Attacke“. Ein weiteres Kriterium kann die Formation des Angriffs sein: Ist es ein direkter Angriff zwischen zwei Beteiligten oder spielt eine dritte Instanz eine Rolle? Im Web-Bereich kommt hier die Schwierigkeit hinzu, dass sich Angriffe meist

zwischen zwei direkt Beteiligten abspielen, Ausgang und Zielpunkt des Angriffs jedoch eine dritte Instanz ist.

2.1.2 Eine Übersicht der häufigsten Attacken auf Applikationen

Mittlerweile kennt hoffentlich jeder Softwareentwickler, der Web-Applikationen entwickelt, das Open Web Application Security Project (OWASP, www.owasp.org) und dessen exzellenten „Guide to building secure web applications". Eine andere sehr gute Quelle von Erklärungen, wie Web-Attacken eigentlich funktionieren, findet man bei [Hus]. OWASP veröffentlicht regelmäßig eine Liste der Top-Ten-Angriffsformen auf Web-Applikationen, die wir kurz analysieren wollen:

A1 Unvalidated Input,
A2 Broken Access Control,
A3 Broken Authentication and Session Management,
A4 Cross Site Scripting,
A5 Buffer Overflow,
A6 Injection Flaws,
A7 Improper Error Handling,
A8 Insecure Storage,
A9 Application Denial of Service,
A10 Insecure Configuration Management.

Angeblich richten sich 70% der Attacken auf Applikationen und nur wenige auf Betriebssysteme oder Netzwerkkomponenten – ein klarer Hinweis auf die Schwächen der Applikationssoftware in Bezug auf Security. Sollten in dieser Liste nicht auch Phishing-Attacken auftauchen? Normalerweise sind dies semantische Attacken auf Client-Seite. Die Frage ist nur, wie weit serverseitig diese Attacken erleichtert werden. Dazu sehen wir weiter unten ein Beispiel, in dem Cross-Site-Scripting zu leichteren Durchführbarkeit von Phishing-Attacken führt. Interessant sind auch „Meta-Attacken" wie das sog. *War-Googling*, d. h. die Verwendung von Google zum Finden von ungeschützten URLs auf gefährliche Administrationsfunktionen in Application-Servern wie z. B. die JMX-Console (s. u.). Lassen Sie uns die zehn Angriffsformen nun grob in drei Kategorien aufteilen:

Input-/Output-bezogene Angriffe:
A1 Unvalidated Input,
A4 Cross Site Scripting,
A5 Buffer Overflow,
A6 Injection Flaws,
A7 Improper Error Handling,
A9 Application Denial of Service.

Infrastrukturbezogene Angriffe:
A8 Insecure Storage,
A9 Application Denial of Service,
A10 Insecure Configuration Management,
AAA (Authentication, Authorization, Accounting)-bezogene Angriffe,
A2 Broken Access Control,
A3 Broken Authentication and Session Management,
A9 Application Denial of Service.

Offensichtlich machen die Input-/Output-bezogenen Schwächen den größten Teil der Attacken aus. Man sollte also vermuten können, dass dieser Tatsache softwaretechnisch Rechnung getragen wird und jedes Web-Framework mit einem ausgezeichneten Sub-Framework für sämtliche Fälle von Input-/Output-Validierung geliefert wird. Außerdem sollten diese Validation-Frameworks auch separat als intelligente Proxies in der Infrastruktur einsetzbar sein, um eine Defense-in-Depth zu erreichen. Ein Beispiel dafür ist das mod_security-Modul für den Apache-Webserver. Trotzdem muss konstatiert werden, dass generell die Verfügbarkeit kompletter Frameworks zur Validierung noch sehr mangelhaft ist (siehe dazu auch die schöne Zusammenstellung gängiger Web-Frameworks in Bezug auf die unterstützten Validierungsmethoden in [Vela]. Die Autoren besprechen hier die einzelnen Schwächen und ihre Behebung durch das plattformübergreifende Tool HDIV).

Neben der großen Zahl von Input-/Output-Problemen fällt auch die Rolle der Authentisierung und Autorisierung von Requests ins Auge. Diese unterscheiden sich von den Input-/Output-Problemen dadurch, dass sie nicht nur am Eingang von Firmen oder Applikationen geprüft werden müssen, sondern auch im weiteren Verlauf von Requests durch ein Unternehmensnetzwerk – genauer gesagt, an jeder Trust Boundary im Unternehmen.

Aber reicht es tatsächlich aus, am Eingang des Unternehmens Input/Output zu prüfen und die weiter hinten gelagerten Komponenten auf das Ergebnis der Prüfung vertrauen zu lassen, z. B. die Businesslogik in EJBs? Streng genommen muss jede Komponente nicht nur Authentisierung und Autorisierung prüfen, sondern auch den Input bzw. den Output. Dass dies in vielen Fällen unterlassen wird, ist eine Frage von Performance und Aufwand für die Entwicklung. Anders gesagt: Wenn nur am Eingang kontrolliert wird, dann haben die Filterfunktionen die Aufgabe, den Input wirklich vollständig zu prüfen, also auch gefährliche Zeichen für weiter hinten kommende Komponenten herauszufiltern. Das ist im Grunde genommen ein gefährliches Antipattern, da es Vertrauensbeziehungen schafft, die jederzeit von beiden Seiten ausgehebelt werden können, z. B. durch Änderungen im Source Code der Businesslogik (oder dem Einsatz einer anderen Datenbank), von denen der Filter nichts mitbekommt.

Denial-of-Service-Angriffe auf Application Level stellen ebenfalls ein interessantes Problem dar, da sie in allen Gruppen vertreten sind. Sie werden weiter unten detaillierter besprochen.

2.1.3 Lokale Angriffsformen

Lokale Angriffsformen sind dadurch gekennzeichnet, dass der Angreifer direkten Zugriff auf die angegriffene Maschine bzw. Software bekommt. Dabei verwendete Angriffsformen sind:

- das Ausnutzen von Schwächen in der Erzeugung von Pseudo Random Numbers (PNRG),
- die hostinternen Attacken durch Symbolic Links,
- nicht-atomare Filesystemzugriffe,
- das Ausnutzen von GUI-Nachrichten unter Windows zur Kontrolle anderer Applikationen (Shatter-Attacks),
- Buffer Overflows und
- Installation von Trojanischen Pferden oder Viren.

Bei diesen als „lokale“ Angriffe bezeichneten Angriffen ist der Angreifer als Benutzer des lokalen Systems angemeldet. Die Einordnung der Viren oder Trojanischen Pferde unter die lokalen Angriffe mag überraschen, „fängt“ man sich doch umgangssprachlich diese Formen von Malware im Internet ein. Diese Angriffe werden also zunächst über das Netzwerk ermöglicht, sie korrumpieren jedoch in ihrer Auswirkung das lokale System und damit die hostbasierte Sicherheit. Diese Konfusion lässt sich auch marketingtechnisch gut ausnutzen. So versprechen manche Hersteller von Antivirensoftware die „Sicherheit des Internet“ verbessern zu wollen – gerade so, als ob es für das Internet ein Problem darstellen würde, wenn sich jemand einen Virus oder Trojaner herunter lädt oder auf eine Phishing-Attacke hereinfällt.

Gemeint ist häufig wohl stattdessen die Sicherheit *im* Internet, das heißt, während man online ist. Aber auch hier zeigt sich, dass die Angriffe, die sich speziell auf die Verbindung beziehen – also die Tatsache, dass ein User über das Internet mit einem Server verbunden ist und jemand die Daten mitlesen oder verändern kann – eher in der Minderzahl sind. So konnte ein populärer Dienstanbieter wie E-Bay noch bis vor kurzem sein Geschäft überwiegend ohne Kanalverschlüsselung und damit ohne Schutz der Integrität und Vertraulichkeit anbieten. Für die heute gängigen Attacken muss man daher schon vom Bedrohungsmodell der verteilten Systeme ausgehen, das bedeutet sich gegenseitig misstrauende Teilnehmer (Rechner und Benutzer).

2.1.4 Zeitliche Entwicklung der Attacken

Greift man nur die Attacken auf Authentisierung bzw. deren Credentials heraus und betrachtet deren zeitliche Entwicklung (siehe Abb. 2.1), so ergibt sich eine „Leiter“ aus eingesetzter Sicherheitstechnologie und den jeweiligen Attacken: Am Anfang stand das Multi-User-System, dessen Sicherheit meist durch verschiedene Passwörter der Benutzer erreicht wurde. Bereits damals gab es Attacken auf die

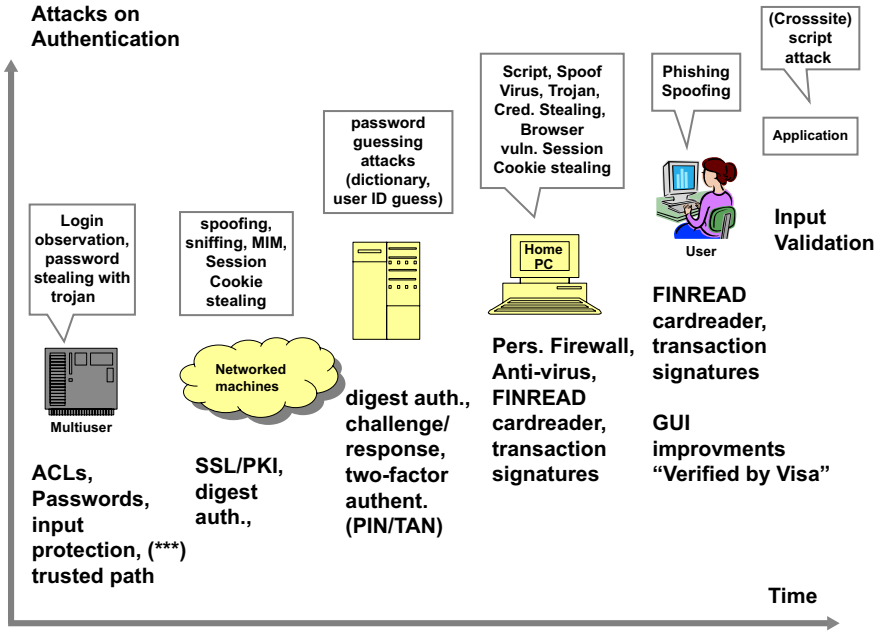

Abb. 2.1 Hierarchie der Authentisierungsmethoden und daraus resultierende Attacken

Credentials in Form von Trojanern oder dem schlichten Mitlesen über der Schulter des betroffenen Users – was aus den heute überall implementierten „*****“ als Echo der für das Passwort eingetippten Zeichen resultierte (eine Kritik daran sowie grundsätzliche Vorschläge zur Verbesserung der Passwortdialoge finden sich bei Peter Gutmann, Security Usability Fundamentals, [Gutm] S. 111ff.). Die Vernetzung der Rechner brachte völlig neue Gefahrenpunkte. Auf das Ablauschen von Daten auf dem Transportweg war die Antwort SSL. Mögliche Man-in-the-middle (MITM) Attacken glaubte man durch das Konzept der Zertifikate sowie durch vertrauenswürdige Autoritäten zu deren Erstellung in den Griff zu bekommen. Mit der zunehmenden Absicherung der Transportwege geriet die Authentisierung auf Seiten der Server mehr in den Blickpunkt: Auf die Schwäche der Passwörter antwortete man mit Multi-Faktor-Authentisierungen mittels PIN/TAN oder Handy. Als die serverseitige Authentisierung sicherer wurde, begann der Angriff auf die Plattform der User: den PC. Trojanische Pferde, die Credentials wie PIN/TAN lokal ausforschen und weiterschicken, können mithilfe von Viruscheckern nur eingeschränkt bekämpft werden. Auf die semantischen Attacken in Form von Phishing waren die Antwort dynamische Challenge-Response-Verfahren mithilfe von kleinen Rechnern, die die Challenges zugesandt bekommen und Responses errechnen, unabhängig von der unsicheren PC-Plattform. Dynamische Man-in-the-Middle-Attacken sowie Trojanische Pferde, die Keyboard-Logging etc. durchführen können, erfordern jedoch eigentlich Transaktionssignierung durch sichere Smartcard-Systeme auf Seite der Kunden – was jedoch die meisten Firmen

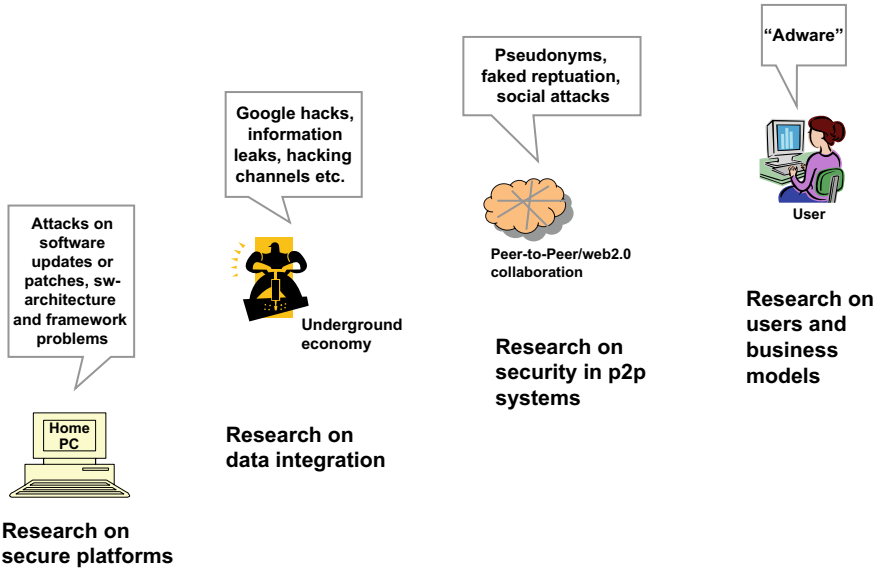

Abb. 2.2 Neue Angriffsformen und die Gegenmaßnahmen

aufgrund der Kosten sowie erschreckend schlechter Usability [Gutm] scheuen. Mit den Cross-Site-Scripting (XSS) Attacken geriet wieder der Server bzw. diesmal die Applikation auf dem Server in den Blickpunkt der Angreifer. Jetzt werden Schwächen der Input-Validierung der Applikation dazu benutzt, die Credentials eines Benutzers zu stehlen oder in dessen Namen Bestellungen zu tätigen oder generell Effekte zu erzielen (z. B. das Rücksetzen von Geräten auf den unsicheren Auslieferungszustand).

Aber auch außerhalb des Bereichs der Authentisierung gibt es genügend Beispiele für Paare aus Attacke und Abwehr. Abbildung 2.2 zeigt beispielsweise Buffer Overflows und als Antwort der Betriebssysteme Non-Executable-Stacks, die Randomisierung des Adressraums oder der Einsatz von Magic Fields (siehe dazu den Abschnitt zu Buffer Overflows weiter unten). Welche Angriffsformen (lokal oder über das Netzwerk) sind nun gefährlicher? Die Antwort darauf erweist sich als erstaunlich schwierig. Der momentane Trend geht jedoch eindeutig in Richtung semantischer bzw. sozialer Attacken, wobei auch die „klassischen" Attacken, z. B. auf die Plattformsicherheit, sich weiter ausdehnen werden. Leider sind hierzu die Abwehrmaßnahmen noch weit weniger entwickelt als z. B. bei der Übertragung von Daten über das Internet.

Bei der Betrachtung der Angriffsformen fallen einige Dinge ins Auge: Die Angriffsziele wechseln und kehren auch wieder zurück, dann allerdings mit neuen Angriffstechniken. Die Angriffstechniken selber werden zunehmend kombiniert (Phishing mit XSS-Attacken). Die Abwehrmaßnahmen sind den Angriffen nie weit voraus und vor allem: Sie beseitigen häufig die Schwächen nicht wirklich. So ist die Gefahr von Man-In-The-Middle-Attacken längst nicht durch den Einsatz

von SSL gebannt, da viele User die Konzepte hinter Zertifikaten und CAs nicht verstehen. Manche Abwehrmaßnahmen können sogar neue Angriffsvektoren schaffen oder sie verwenden, z. B. problematische visuelle Marker, deren schlechte Usability letztlich nicht für mehr Sicherheit sorgt.

An den Attacken zeigt sich sehr deutlich, dass von Sicherheit nur noch im Zusammenhang des Gesamtsystems, bestehend aus den beteiligten Plattformen, den Netzwerken sowie den unterschiedlichen Akteuren, gesprochen werden kann. Es gibt nicht mehr den einzelnen Angriffspunkt oder die einzelne Angriffstechnik. Stattdessen muss sich eine IT-Lösung heute in dem genannten Systemzusammenhang bewähren, wenn sie sicher genannt werden will.

2.2 Die Attacken im Einzelnen

Hier können wir uns auf eine kurze Beschreibung der wesentlichen Attacken beschränken. Für Web-Applikationen sind ausgezeichnete Informationen im bereits erwähnten Web Application Security Guide unter www.owasp.org verfügbar. Detaillierte Beschreibungen finden sich ebenso im Buch zur Sicherheit von Web-Applikationen von [Hus]. Aktuelle Arbeiten zu Angriffen auf allen Ebenen finden sich auch auf der Webseite zum Buch.[2]

2.2.1 Externe Attacken auf Applikationslevel

Cross-Site-Scripting (XSS) Attacken

Cross-Site-Scripting-Attacken sind in mehrfacher Hinsicht interessant. Sie sind einmal ein Angriff über einen gutgläubigen Dritten, mit dem das Opfer eine Vertrauensbeziehung hat. Es gibt sie in persistenter Form (der eingeschleuste Code wird beim Server abgespeichert und später an andere ausgeliefert, z. B. als Profildaten eines Users) wie auch in nicht-persistenter Form (der Server prüft eine Eingabe, die Code enthält, nicht richtig, sondern sendet die Daten an das Opfer).

Cross-Site-Scripting-Attacken sind deshalb so gefährlich, weil sie das Vertrauensverhältnis, das ein Nutzer mit einem Server unterhält, ausnutzen und missbrauchen. Für den Intranetprogrammierer kommen diese Attacken häufig völlig überraschend, weil bösartige Nutzer im „User Conceptual Model“ eines Intranetentwicklers nicht vorkommen. Die Möglichkeit von Cross-Site-Scripting-Attacken machen somit noch einmal deutlich, dass im momentanen Sicherheitsmodell für das Web die Server größtenteils die Verantwortung für die Sicherheit des Clients übernehmen müssen. Abbildung 2.3 zeigt die prinzipielle Funktionsweise einer Cross-Site-Scripting-Attacke (gezeigt ist die nicht-persistente Form des Angriffs).

---

[2] www.kriha.de/krihaorg/sicheresysteme.html