

1 Einleitung

1.1 Über dieses Buch

Als im Herbst 2004 das Konzept für die erste Auflage dieses Buchs erstellt wurde, hatte PHP bereits viele andere Web-Skriptsprachen hinter sich gelassen. Millionen von Websites bauen auf die dynamische Skriptsprache, und von einfachen Gästebüchern bis hoch komplexen mehrstufigen Business-Applikationen sind PHP-Anwendungen in allen Größen, Farben und Formen zu finden.

PHP 5 bietet all denen, die zuvor die etwas lückenhafte Implementierung objektorientierter Programmierung von ernsthaften Gehversuchen abgehalten hat, eine umfassendere OOP-Schnittstelle und führte dazu, dass noch mehr großen Projekten der Sprung von Lösungen mit Java oder .NET zu der freien Skriptsprache gelang.

Mit dem immensen Wachstum der Nutzer- und Entwicklergemeinde wuchsen allerdings auch die Schwierigkeiten. Waren sicherheitsrelevante Fehler in PHP-Programmen schon seit den Zeiten von PHP 3 ein ernst zu nehmendes Problem, so kam es im Laufe des Jahres 2004 zu einigen mehr oder weniger spektakulären Sicherheitslücken. Die Forensoftware phpBB war besonders schlimm betroffen, ist sie doch (aufgrund ihres reichhaltigen Feature-Umfangs und der relativ einfachen Administration) eine der Killer-Applikationen für PHP. Ohne freie und leicht zu benutzende Produkte wie phpBB wäre die Verbreitung von PHP sicher nicht so schnell gegangen – allerdings auch nicht die Verbreitung des ersten PHP-Wurms Santy¹, der eine Lücke in der Forensoftware ausnutzt, um Schadcode zu laden und auszuführen. Ironischerweise ist Santy ausgerechnet in Perl geschrieben, lange Zeit die Sprache, an der sich PHP messen musste.

1. <http://www.viruslist.com/en/viruses/encyclopedia?virusid=68388>

Andere Sicherheitsprobleme auf Internetseiten mit dem Label »PHP powered« beschäftigten gleich die Staatsanwaltschaften² oder zahlreiche Mitarbeiter großer Telekommunikationskonzerne³. Bei allen Fehlern handelte es sich um Programmier- oder Designschnitzer, die vermeidbar gewesen wären. Obwohl auch in PHP selber Bugs entdeckt (und behoben) wurden, die von einem böswilligen Angreifer für seine finsternen Zwecke hätten verwendet werden können, liegt die große Mehrzahl der für die erfolgreiche Installation von Rootkits, Backdoors oder andere Exploits verantwortlichen Sicherheitslücken in den Anwendungen selber.

PHP ist den Kinderschuhen entwachsen, darüber besteht weitgehend Einigkeit. Anders als bei konventionellen Programmiersprachen wird allerdings in der Netz-Öffentlichkeit die Qualität einer Sprache oft an der Qualität der Anwendungen gemessen. So hat das Website-Management-System PHP-Nuke dem Ruf der Sprache PHP durch seine zahlreichen Sicherheitslücken nicht unerheblich geschadet – ähnlich wie das berüchtigte `formmail.pl` in Perl das sprichwörtliche Negativbeispiel ist. Das scheint ungerecht, schließlich wird C++ auch nicht für Fehler in Windows verantwortlich gemacht oder Assembler für Bugs im Interrupt-Handling des Linux-Kernels beschuldigt. Dennoch hat diese Haltung auch Vorteile: Sie zwingt das PHP-Team (die »PHP Group«) mittelfristig, »Best Practices« zu entwickeln und PHP-Programmierer zu mehr Sicherheitsbewusstsein zu erziehen. Dieses Buch soll dabei helfen, indem es Problemfelder aufzeigt, Lösungsmöglichkeiten anbietet und anhand klarer Checklisten die Absicherung neuer und vorhandener Skripte erleichtert.

An wen richtet sich dieses Buch?

Das Buch »PHP-Sicherheit« wurde mit Blick auf zwei Gruppen von Lesern geschrieben: Fortgeschrittene und Profis mit Wartungsaufgaben sowie Systemadministratoren für Web- und Datenbankserver.

PHP-Neulinge

Gerade Neulinge auf dem Feld der PHP-Entwicklung gehen – das haben die Autoren zumindest in vielen Fällen beobachten können – mit einer erfrischenden Naivität an die Programmierung ihrer ersten dynamischen Webseite heran. Das ist nicht grundsätzlich falsch, schließlich ist das Prinzip von »Trial and Error« so alt wie die Programmierung selber. Da aber PHP eine serverbasierte Skriptsprache ist und die Entwicklung direkt auf dem Webserver stattfindet, sind fehlerhafte »Hallo Welt«-Elaborate nicht nur für ihren Schöpfer peinlich,

2. <http://www.heise.de/newsticker/meldung/50516>

3. <http://www.heise.de/newsticker/meldung/55057>

sondern mit etwas Pech auch eine große Gefahr für andere Server im Internet.

Dieses Buch soll dazu dienen, Einsteiger auf Gefahren und Risiken aufmerksam zu machen, die sie bei der Entwicklung und beim Einsatz von PHP-Skripten beachten müssen. Der Großteil aller Anfängerfehler lässt sich mit einigen grundlegenden Verhaltens- und Implementierungsregeln vermeiden – und damit selbst das persönliche Gästebuch sicherer gestalten.

Auch fortgeschrittene PHP-Entwickler, die bereits mittlere bis große Anwendungen selber oder als Teil eines Teams entwickelt haben, stehen oft vor ähnlichen Problemen. Sogenannter »Legacy Code«, also gewachsene Produkte, die ursprünglich einen ganz anderen Zweck erfüllten, strotzen oft nur so vor Sicherheitsmängeln oder Designfehlern. Und doch ist ein kompletter Rewrite oft nicht machbar – also steht eine Sicherheitsüberprüfung des gesamten Quellcodes an. Die Checklisten im Anhang helfen, diese Aufgabe zu systematisieren – und selbst der versierteste PHP-Crack wird einige der in diesem Buch vorgestellten Lücken nicht kennen.

*Fortgeschrittene
PHP-Entwickler*

Der letzte Teil des Buches richtet sich hingegen an Systemadministratoren, also diejenigen, die unter schlecht geschriebenen und schlampig gewarteten Programmpaketen leiden und nach einem erfolgreichen oder misslungenen Hack die Aufräumarbeiten erledigen müssen. Obwohl sie meist Zugriff auf die Anwendung haben, können oder dürfen die Administratoren nur selten selber sicherheitskritische Änderungen durchführen. Um Schaden abzuwenden, müssen also die Webserver so konfiguriert werden, dass ein Angreifer selbst ein sehr unsicheres Skript nicht für seine Zwecke missbrauchen kann – der Webserver muss gegen Angriffe abgehärtet (»hardened«) werden.

Systemadministratoren

1.2 Was ist Sicherheit?

In der IT existiert ein geflügelter Spruch, der besagt, dass Daten erst dann sicher seien, wenn sie in einem Safe an einer unbekanntem Stelle auf dem Meeresboden versenkt würden. So übertrieben das auch klingen mag, dieses Sprichwort enthält einen Funken Wahrheit. Absolute Sicherheit kann (ohne Tresore und Tiefsee-Lagerung in Betracht zu ziehen) nicht erzielt werden, alle Bemühungen müssen stets als »best effort« gelten.

Eine sinnvolle Definition des Sicherheitsbegriffes kann stets nur kontextbezogen gefunden werden. Institutionen wie Finanz- oder Meldeämter, die mit hoch- und höchstvertraulichen Daten zu tun haben, werden unter Sicherheit etwas völlig anderes verstehen als jemand, der

auf seiner privaten Homepage in einem einfachen CMS Bilder seines Goldfisches ausstellt. Daher kann man die Sicherheit von webbasierten Systemen (um die es in diesem Buch letztlich gehen soll) folgendermaßen umschreiben:

Ein System kann als sicher gelten, wenn die Kosten für einen erfolgreichen Angriff den möglichen Nutzen übersteigen.

Cracker und sonstige böse Buben verfügen nämlich nicht über unbegrenzte Zeit und Mittel. Die erste Gruppe von Angreifern, »Scriptkiddies«, rekrutiert sich überwiegend aus Jugendlichen und jungen Erwachsenen mit wenigen oder keinen Fachkenntnissen. Das typische Scriptkiddy verfügt über eine gut sortierte Bibliothek vorgefertigter Angriffstools für viele verschiedene Lücken und sucht sich seine Opfer meist anhand vorhandener Lücken aus. Derartige Angreifer werden sich meist leichte Ziele aussuchen, die gleichzeitig vielversprechende Möglichkeiten bieten. Ein Webserver, der schnell ans Internet angebunden ist und auf dem eine uralte Version des Forums phpBB verwendet wird, ist verlockende Beute für Scriptkiddies. Er ist leicht zu »knacken« und kann dann als Ablageplatz für Raubkopien oder als Relay, also Zwischenstation, für weitere Angriffe missbraucht werden.

Ist aber die verwendete Software auf dem neuesten Stand oder auch recht unbekannt, wird mehr Aufwand notwendig, um in den Server einzudringen – ein Einbruch lohnt sich oft nicht mehr, die meisten Scriptkiddies werden schnell von einem Server ablassen, der auf den ersten – meist automatisierten – Blick keine Angriffsfläche bietet.

Ein Unternehmen, das sehr viele wertvolle Daten verwaltet, ist jedoch auch für erfahrenere Cracker, die ihre Raubzüge aus kommerziellen Gründen durchführen, ein sehr attraktives Ziel. Hacker werden viel Zeit und Mittel aufwenden, um an diese Daten zu gelangen, es reicht daher nicht, die Datei mit sämtlichen Kundendaten in einem versteckten, aber für den Webserver zugänglichen Verzeichnis abzulegen. Sobald die möglichen Eindringlinge das Ziel als sehr attraktiv eingestuft haben, werden sie sich so lange an den frei zugänglichen Teilen verbeißen, bis sie einen Zugang finden. Mit einer Kundendatenbank voller Kontoinformationen können sie schließlich wesentlich mehr anfangen als mit den Bildern einer erfolgreichen Goldfischzucht.

1.3 Wichtige Begriffe

Um in den folgenden Kapiteln nicht stets neue Erklärungen finden zu müssen, möchten wir einige wichtige Begriffe aus dem Sicherheitsjargon vorab erklären.

Defense in Depth

Werden Sicherheitskonzepte angesprochen, taucht das Schlagwort »Defense in Depth« immer öfter auf, um ein möglichst schlagkräftiges Sicherheitsprinzip zu beschreiben. Und tatsächlich ist das Prinzip der »Tiefenverteidigung«, wie eine deutsche Übersetzung des Begriffes lauten könnte, bei konsequenter Umsetzung sehr wirksam.

Der von Defense in Depth verfolgte Ansatz geht von einer zwiebelschalenförmigen Sicherung aus, bei der eine Anwendung durch Sicherheitsmaßnahmen auf mehreren Ebenen geschützt ist. Diese Sicherung beginnt bereits auf der Netzwerkebene: Firewalls trennen die Web-Infrastruktur vom internen Netzwerk oder VPN des Betreibers, und Paketfilter sorgen dafür, dass nur diejenigen Benutzer Zugriff auf sensitive Bereiche haben, die auch administrativ dafür zugelassen sind.

Auf Betriebssystemebene setzt die zweite Verteidigungslinie gegen Cracker und sonstige Finsterlinge an: Ein gehärteter Linux-Kernel wehrt selbsttätig Angriffe gegen den IP-Stack oder interne Funktionen ab und erlaubt kein Nachladen und Ausführen von Kernel-Modulen. Eine im Kernel implementierte Changeroot-Umgebung (chroot) separiert Anwendungen voneinander – und weitere Maßnahmen, die auf Betriebssystemebene implementiert werden.

Betriebssystem

Eine Web Application Firewall (WAF), die webbasierte Angriffe aus dem produktiven Traffic herausfiltert und die Administratoren informiert, ist die zweite Verteidigungsebene.

Web Application Firewall

Bei Webapplikationen ist die nächste Ebene einer Defense-in-Depth-Strategie der Webserver, der im Rahmen seiner Möglichkeiten gehärtet werden sollte. Dazu gehören nicht nur Techniken zur Vermeidung von Informationslecks, sondern auch Sicherheitsmaßnahmen wie der Einsatz von SSL. Auch das Apache-Modul `mod_security`, das sich selber bereits als WAF bezeichnet, oder `chroot`- und `suExec`-Mechanismen sind Teil dieser Ebene.

Webserver

Die nächste Ebene des Defense in Depth sind die PHP-Anwendungen selber, und damit der Hauptgegenstand dieses Buches. Auch diese Anwendungen lassen sich wiederum in Schichten aufteilen, die separat voneinander gesichert werden.

PHP-Anwendungen

Ein-/Ausgabe

Auf der Datenebene sollte gesichert sein, dass keine schadhaften oder in böswilliger Absicht erstellten Daten (wie XSS-Angriffe zweiter Ordnung, siehe den nächsten Abschnitt) in die Datenspeicher gelangen können. Die Datenabfrageschicht sollte so implementiert sein, dass Angreifer nicht durch Tricks andere Daten als die vom Entwickler vorgesehenen abfragen können – und die Ein-/Ausgabeschicht muss Eingaben auf Schadcode überprüfen, diesen abfangen und möglicherweise unerwünschte Ausgaben verhindern.

Der Vorteil an dieser Sicht der Dinge ist, dass die Zusammenarbeit zwischen den verschiedenen Schichten ähnlich wie im OSI-Modell geregelt ist: Jede Schicht kommuniziert nur mit der ihr direkt vorangehenden und nachfolgenden Ebene. Dadurch können Entwickler und Administratoren mit der größtmöglichen Unabhängigkeit voneinander arbeiten, um ihren jeweiligen Teil der Anwendung abzusichern.

First und Second Order

Bei Angriffen gegen andere Systeme geht man grundsätzlich von zwei verschiedenen Angriffstypen aus, und zwar von direkten Angriffen der ersten Ordnung und indirekten oder Angriffen der zweiten Ordnung.

Angriffe erster Ordnung

Ein *Angriff erster Ordnung* liegt vor, wenn durch eine SQL-Injection, XSS (was dies alles ist, erfahren Sie in den folgenden Kapiteln) oder eine sonstige Attacke direkte Ergebnisse geliefert werden, also etwa die Liste der Administratorpasswörter aus der entsprechenden Datenbanktabelle gelesen und angezeigt wird, oder JavaScript-Code direkt im Kontext der angegriffenen Webseite ausgeführt wird. Angriffe erster Ordnung erlauben dem Angreifer, aufgrund der Antwort des angegriffenen Systems sein Vorgehen zu optimieren und sein Ziel zu erreichen. Dadurch sind First-Order-Angriffe in der Regel leichter durchzuführen als Second-Order-Attacken.

Angriffe zweiter Ordnung

Diese *Angriffe zweiter Ordnung* müssen meist »blind« durchgeführt werden. Der Angreifer weiß oder vermutet, dass an einer bestimmten Stelle in der Zielanwendung nur ungenügende Eingabevalidierung vorgenommen wird, kann diese Tatsache jedoch nicht direkt ausnutzen, weil er an das betroffene Subsystem nicht herankommt. Ein klassisches Beispiel wird im Kapitel 4 »Cross-Site Scripting« behandelt: Log-Dateien oder -Datenbanken sind ein prädestiniertes Ziel für Second-Order-Angriffe. Der Angreifer sorgt dafür, dass mit Schadcode versehene Anfragen gespeichert werden, und wartet nun darauf, dass ein Administrator der Zielanwendung diese Daten auswertet. Ob und wann das geschehen wird und ob der Angriff dann auch erfolgreich sein wird, kann er nur selten vorhersagen.

Somit sind Second-Order-Angriffe häufig wesentlich komplizierter und langwieriger in der Durchführung, was dazu führt, dass Entwickler und Administratoren ihnen nur geringe Bedeutung beimessen. Die Tatsache, dass sie jedoch meist direkt gegen Inhaber hoch privilegierter Accounts gerichtet sind, macht diese Angriffsklasse für Angreifer wie Verteidiger gleichermaßen zu einer lohnenden Herausforderung.

1.4 Sicherheitskonzepte

Security is a process, not a product.
Bruce Schneier⁴

Seitdem elektronische Datenverarbeitung existiert, haben sich kluge Köpfe Gedanken um die Sicherung der Daten und Systeme gegen Missbrauch gemacht. Die resultierenden Sicherheitskonzepte sind oft grundsätzlich fehlerhaft – etwa indem sie nur zur Zeit der Erstellung aktuelle Technologien berücksichtigen oder sich auf die Geheimhaltung bestimmter Daten verlassen. Beides hat sich in der Vergangenheit oft als Trugschluss erwiesen.

Auch Verschlüsselungsalgorithmen oder Verfahren zur Erhöhung der Sicherheit sind nicht dadurch vertrauenswürdig, dass sie von ihrem Entwickler geheim gehalten werden. Obwohl die Mechanismen zur Passwortverschlüsselung z.B. bei Microsoft Windows nicht öffentlich zugänglich waren, konnten sie doch überwunden werden.

Verschlüsselung

Der lange Zeit als unüberwindbar geltende RC5-64-Verschlüsselungsalgorithmus wurde von einem Projekt zehntausender Anwender auf der ganzen Welt in gut vier Jahren geknackt. Laut Moores Gesetz verdoppelt sich die Leistung der jeweils aktuellen Prozessorgeneration immer noch alle 18 Monate.

Prüfsummen, die mit den Algorithmen SHA-1 oder MD5 erstellt wurden, lassen sich inzwischen in endlicher Zeit durch sogenannte »Kollisionen«, also die Erstellung einer identischen Prüfsumme für zwei verschiedene Ausgangswerte, überlisten⁵.

Prüfsummen

Daher sollte ein Sicherheitskonzept nicht daraus bestehen, sich auf die Vertraulichkeit der eingesetzten Werkzeuge zu verlassen oder darauf zu vertrauen, dass die für einen Angreifer verfügbare Rechenleistung nicht ausreichen wird, um Ihren Verschlüsselungsalgorithmus zu brechen. Mit Seiten wie [passcracking.com](http://www.passcracking.com)⁶ und neuartigen zeitsparen-

4. <http://www.schneier.com/>

5. http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

6. <http://www.passcracking.com>

den Brute-Force-Verfahren sind selbst MD5-Passwörter in einer recht kurzen Zeit zu knacken.

Um zu einem tragfähigen Sicherheitskonzept für Ihre Firma oder auch nur Ihre privat entwickelte PHP-Applikation zu gelangen, sind einige Schritte notwendig. Die tatsächliche Absicherung Ihrer PHP-Skripte ist nur einer dieser Schritte, wenn auch der wichtigste.

Alle Dienste absichern

Betreiben Sie eine dynamische Website auf einem eigenen Server, müssen Sie (oder Ihre Mitarbeiter) dafür sorgen, dass nicht nur die verwendeten PHP-Anwendungen sicher sind, sondern auch dass alle anderen von außen erreichbaren Dienste keine Angreifer hereinlassen. Ihre Systeme sind nur so sicher wie das schwächste Glied in der Kette – in vielen Fällen ist das jedoch mit heißer Nadel gestrickte PHP-Software. Einen kompletten Leitfaden zur Absicherung Ihrer Serversysteme zu schreiben, würde den Rahmen dieses Buches sprengen – wir werden uns hauptsächlich mit PHP und einigen von PHP verwendeten Subsystemen wie Web- und Datenbankservern beschäftigen.

In einem Unternehmen, das vertrauliche Informationen behandelt, muss ein Sicherheitskonzept integraler Bestandteil der Firmenprozesse und -politik sein. Datenverluste oder – schlimmer noch – Datendiebstähle gehören im Informationszeitalter zu den unangenehmsten Vorkommnissen für jede Firma.

ISO 17799

Ein tragfähiges Sicherheitskonzept ist sehr umfangreich und kann nicht auf wenige Punkte reduziert werden. Ein guter Anhaltspunkt ist die internationale Richtlinie ISO 17799, »Code of Practice for Information Security Management«, die als industrieweit anerkannter Standard für den sicheren Umgang mit IT-Systemen und allen dazugehörigen Subsystemen gilt. Wie alle ISO-Standards ist die Richtlinie leider nicht kostenlos erhältlich, kann aber direkt bei der ISO bestellt werden. Obgleich in ISO 17799 (die in vielen Unternehmen als BS 7799, kurz für »British Standard«, bekannt ist) kein Bezug auf webbasierte Systeme genommen wird, enthält die Richtlinie viele wichtige Anregungen, die Sie benutzen können, um die sicherheitsrelevanten Abläufe in Ihrem Unternehmen zu verbessern.

Im nächsten Abschnitt fassen wir die wichtigsten Punkte kurz für Sie zusammen und erläutern den Zusammenhang mit PHP-Sicherheit.

1.5 ISO 17799

Besonders in großen Unternehmen ist die Sicherheit der IT-Infrastruktur und ganz besonders der unternehmensinternen Daten eines der wichtigsten Ziele. Um Abläufe zu standardisieren und das Handling sicherheitsrelevanter Prozesse auf einen einheitlichen Nenner zu bringen, wurde von der British Standards Institution (BSI, nicht zu verwechseln mit dem deutschen Bundesamt für Sicherheit in der Informationstechnik) der Standard 7799 ins Leben gerufen. In diesem Dokument werden ausführlich »Best Practices«, also als vorbildlich anerkannte Abläufe für die Sicherheit in Informationssystemen aufgeführt. Neben Aspekten wie der physischen Zugangssicherung enthält das Standardwerk, das mittlerweile als internationaler Standard ISO 17799 aufgelegt wurde, auch für Webanwendungen wichtige Punkte.

Die ISO unterteilt Sicherheit in Informationssystemen in drei Faktoren, die in einem sicheren System stets erfüllt sein sollten:

- *Vertraulichkeit*, also die Sicherheit, dass Information nur dem zugänglich ist, der über die notwendige Autorisierung verfügt.
- *Integrität* – Informationen und informationsverarbeitende Vorgänge müssen stets akkurat und vollständig sein.
- Die ständige *Verfügbarkeit* aller Informationssysteme für berechtigte Benutzer ist der dritte wichtige Faktor.

Der Standard schreibt vor, dass alle Mechanismen, die zur Wahrung dieser Faktoren etabliert werden, regelmäßig kontrolliert werden sollen, um auch auf neue Anforderungen reagieren zu können. Das werden die meisten Administratoren von Webservern intuitiv beherzigen – ist doch die Kontrolle auf neue Software-Updates nichts anderes, als der Standard fordert.

Auch auf die Absicherung bei »third-party access«, also dem Zugriff Dritter auf die eigene Infrastruktur, legt ISO 17799 Wert. In der PHP-Welt ist das beispielsweise ein API Ihrer Anwendung, an die Ihre Kunden ihre eigenen Anwendungen anschließen können, um Funktionen und Informationen mit Ihrem System auszutauschen. Hier sollten Vereinbarungen vor Missbrauch schützen.

Auch die Sicherheit von Zugangskennungen wird behandelt – dem Standard folgend, sollten Sie Ihren Benutzern – sei es in einem PHP-Forum, einem CMS oder einer Administrationsoberfläche – nur exakt die Privilegien geben, die für die Ausführung der jeweiligen Aufgabe notwendig sind, und darauf achten, dass Passwörter sicher sind und geheim bleiben. Temporäre Passwörter, die direkt nach der Registrierung oder auf Anforderung durch den Benutzer vergeben werden, sollten auf einem sicheren Weg (nicht im Klartext per E-Mail, wie leider

meist üblich, sondern möglichst per SSL-geschützter HTTP-Verbindung) übergeben und vom Benutzer sofort geändert werden.

Eine vollständige Besprechung der ISO 17799 bzw. der Nachfolger dieser Richtlinie würde den Rahmen dieses Buches sprengen. Sie ist jedoch als Zusammenfassung der Vorgänge und Arbeitsweisen, die guten Systemadministratoren in Fleisch und Blut übergegangen sind, eine wertvolle Hilfe. ISO 17799 kann kostenpflichtig im Internet direkt über den Online-Shop der ISO⁷ bestellt werden.

1.6 Wie verkaufe ich Sicherheit?

Wenn Sie dieses Buch lesen, sind Sie vielleicht auf irgendeine Weise professioneller PHP-Entwickler – entweder als freiberuflicher Programmierer oder als Angestellter in einem Unternehmen. Möchten Sie Ihre bestehenden PHP-Anwendungen sichern oder eine neue Anwendung mit besonderem Augenmerk auf die Sicherheit entwickeln, werden Sie feststellen, dass dies mit höherem Aufwand verbunden ist, als das Skript einfach »herunterzuhacken«. Ihr Auftrag- oder Arbeitgeber muss diesen Aufwand finanzieren, und Sie müssen begründen können, warum diese Zusatzkosten zwingend notwendig sind.

Was für Sie völlig einleuchtend sein dürfte – schließlich hätten Sie sonst nicht dieses Buch gekauft –, ist Managern oder Projektleitern oft leider nicht ganz so leicht zu vermitteln.

Insbesondere in Unternehmen, deren Hauptgeschäftsfeld nicht die IT ist, ist die »Security Awareness«, also das Sicherheitsbewusstsein, oft nur ungenügend ausgeprägt. Das äußert sich in Problemen wie per Post-It-Haftnotiz am Monitor befestigten Passwörtern, aber auch in einer gewissen Naivität für Applikationssicherheit.

Für Manager zählt oft hauptsächlich, wie sich eine Ausgabe rechnet: Ergibt sich nach Gegenrechnung aller Kosten noch immer ein Gewinn für das Unternehmen, sind Ausgaben leicht zu vermitteln. Sicherheit bringt allerdings direkt keine zusätzlichen Umsätze, ist also zunächst als Kostenfaktor ohne Gewinn einzustufen.

*Sicherheit ist ein
Produktmerkmal.*

Genauer betrachtet, stimmt das natürlich nicht – das müssen Sie dem Management vermitteln. Entwickeln Sie ein Softwareprodukt, das später von Ihrer Firma verkauft oder lizenziert werden soll, ist Sicherheit ein wichtiges Produktmerkmal, das die Verkäufe äußerst positiv beeinflussen kann. Setzen Sie webbasierte Anwendungen für das eigene Unternehmen ein, können Sie Datensicherheit nur dann

7. <http://www.iso.org/>

gewährleisten, wenn die Anwendung gegen Diebstahl von Sessions, SQL-Injection und XSS abgeschottet ist.

Der Verlust von Kundendaten kann ernste juristische Konsequenzen nach sich ziehen und zum Ruin Ihres Unternehmens führen. So musste das amerikanische Unternehmen CardSystems im Juni 2005 zugeben, dass die Kreditkartendaten von über 40 Millionen Kunden durch Hacker gestohlen worden waren. Die Firma führte als sogenannter »Third Party Processor« Kreditkarten-Transaktionen im Auftrag von Händlern durch und leitete diese an die Kreditkarten-Unternehmen weiter.

Beispiel CardSystems

Da der Diebstahl durch eine Sicherheitslücke im Netzwerk des Unternehmens und fahrlässiges Verhalten einiger Mitarbeiter möglich wurde, kündigten daraufhin einige Kreditkartenfirmen ihre Verträge mit CardSystems und entzogen der Firma dadurch die Grundlage ihrer Dienstleistungen.

Bei webbasierten Anwendungen ist ein Beispiel für den durch Sicherheitslücken entstehenden Vertrauensverlust das Portalsystem phpNuke. Obgleich die Idee und der Funktionsumfang von phpNuke prinzipiell sehr nützlich sind, hat die Menge an konzeptbedingten Sicherheitslücken das Open-Source-Projekt so weit diskreditiert, dass man von einer Installation inzwischen nur noch abraten kann. Auch das freie Forensystem phpBB hat in letzter Zeit durch viele kritische Sicherheitslücken, die unter anderem den Wurm »Santy« ermöglichten, von sich reden gemacht – einige Hosting-Firmen verbieten in der Konsequenz den Einsatz dieses Forums auf ihren Servern.

Beispiel phpNuke

Verdient Ihr Unternehmen sein Geld mit der Entwicklung und dem Verkauf webbasierter Applikationen, können solche Boykottaktionen empfindliche Umsatzeinbußen bedeuten – schon eine auf Security-Mailinglisten veröffentlichte kritische Lücke wird viele Administratoren davon abhalten, Kaufempfehlungen für Ihr Produkt auszusprechen. Denn: Wo ein Fehler ist, sind meist noch ein paar ähnliche Schnitzer, und viele Sicherheitslücken beruhen nicht nur auf nachlässiger Programmierung, sondern auf einem fehlerhaften Programmkonzept. Haben Sie Ihren guten Ruf als fähiger Entwickler erst einmal durch einige Sicherheitslücken verspielt, ist es praktisch unmöglich, diesen wiederherzustellen – denken Sie nur an Sendmail, Bind oder wu-ftp, die Musterbeispiele bekannter Open-Source-Produkte mit ellenlanger Fehlerliste.

Neben den direkten Folgen juristischer oder strafrechtlicher Art hat ein »Hack« in einem Ihrer Produkte somit auch verheerende Auswirkungen auf den Ruf Ihrer Firma. Das sollten Sie und auch Ihre Vor-

gesetzten sich stets vor Augen führen, wenn es darum geht, auf Kosten der Sicherheit zu sparen.

Manager-Leitsatz: Sicherheit bedeutet nicht immer mehr Umsatz, aber keine Sicherheit führt zu Umsatzeinbußen!

1.7 Wichtige Informationsquellen

Dieses Buch bemüht sich, Ihnen einen Überblick über die heute bekannten Angriffsklassen bei webbasierten Applikationen zu verschaffen, kann jedoch nie auf dem neuesten Stand sein. Serveradministratoren und PHP-Entwickler sollten daher andere Informationsquellen nutzen, um stets »up to date« zu sein. Das betrifft sowohl ganz konkrete Lücken in PHP-Applikationen als auch generelle Techniken und Konzepte. So tauchte etwa im Jahr 2005 die Angriffsklasse »HTTP Response Splitting« erstmals auf, die zuvor völlig unbekannt war und daher auch nur von wenigen Anwendungen abgedeckt wurde. Derlei neuartige Angriffsmuster werden oft zunächst theoretisch in einer Veröffentlichung diskutiert, bevor sie praktisch implementiert und schließlich von Scriptkiddies aus aller Herren Länder ausgenutzt werden.

Informieren Sie sich auf Mailinglisten über aktuelle Security-Trends!

1.7.1 Mailinglisten

Das Gros dieser Diskussionen findet auf den Mailinglisten »BugTraq«, »Full Disclosure« und »webappsec« statt. Insbesondere die ersten beiden Listen gelten als die besten Adressen für die neuesten Exploits und Fehler – jeder Serveradministrator sollte im Grunde gesetzlich verpflichtet werden, mindestens eine der beiden zu abonnieren.

Die übliche Netiquette gilt natürlich auch hier – insbesondere sollten Sie darauf achten, bei Abwesenheit nicht mit einem Autoresponder den mehreren Tausend anwesenden Hackern mitzuteilen, dass Ihre Server momentan leider ungeschützt sind, da Sie im Urlaub weilen. Autoresponder auf »BugTraq« sollen schon für den einen oder anderen Servereinbruch gesorgt haben – und zudem können Sie sicher sein, das Ziel des teilweise recht drastischen Spottes der Liste zu werden.

1.7.2 Full Disclosure

Die Liste »Full Disclosure« ist nicht nur eine Mailingliste, sie steht für eine Art Lebensgefühl in der Security-Gemeinde. Mit »Full Disclosure« wird die Praktik beschrieben, Sicherheitslücken nach Bekanntwerden und Behebung durch den Softwarehersteller mit allen Details zu melden – und zwar eben an die Mailingliste Full Disclosure. Postings an FD, so der Kurzname der Liste, enthalten neben ausführlichen Informationen zu einer gefundenen Sicherheitslücke oft sogenannten »Proof of Concept«-Code, also ein kurzes Skript oder Programm, welches das Vorhandensein der Lücke demonstriert. Da diese Nachweise eines Problems nicht selten den Entwicklern von Würmern, Rootkits oder Exploits als nützliche Vorlage dienen, ist Full Disclosure bei Sicherheitsexperten umstritten. Insbesondere ein großer Softwarehersteller aus Redmond hat sich in der Vergangenheit vehement gegen dieses Vorgehen gewehrt, sei es doch unverantwortlich und führe zu einer massiven Bedrohung der Internet-Infrastruktur. Auch das US-Heimatschutzministerium versuchte in der Vergangenheit, vollständige Veröffentlichung von Lücken zu unterbinden, die US-Copyright-Gesetze im Digital Millenium Copyright Act (DMCA, siehe Glossar) sollten dabei helfen.

Trotz dieser Widrigkeiten hält sich die Praxis der Full Disclosure weiterhin, was die Liste für Systemadministratoren zu einer unentbehrlichen Quelle macht: Zum einen werden Security-Hinweise (die sogenannten »Advisories«) auf keiner anderen Mailingliste so schnell veröffentlicht wie auf FD, und zum anderen kann die tatsächliche Gefahr, die von einer Lücke ausgeht, anhand der Liste gemessen werden. Sobald ein Exploit dort veröffentlicht wurde, können Sie davon ausgehen, dass jeder mit einfachsten Mitteln Angriffe gegen die verwundbare Software starten kann – spätestens dann sollten Sie eine Nachtschicht einlegen, um Ihre Systeme zu patchen.

Da FD nicht moderiert wird, ist die Latenz zwischen Posting und Veröffentlichung zwar sehr kurz, es kommt jedoch fast täglich zu äußerst unangenehmen und ermüdenden Flamewars zwischen den anwesenden Security-Experten, vereinzelt Scriptkiddies und den wie auf jeder großen Liste reichlich vorhandenen Unruhestiftern. Diese Scharmützel ziehen sich teilweise über Tage hin und sorgen für ein sehr schlechtes Signal-Rausch-Verhältnis.

Die Liste Full Disclosure sollte trotzdem Ihre tägliche Pflichtlektüre werden – abonnieren können Sie sie einfach mit einer Mail an die Adresse full-disclosure-request@lists.grok.uk; das Subject sollte »subscribe« (ohne Anführungszeichen) lauten. Alternativ können Sie über das Webinterface⁸ ein Abonnement anfordern.

Der Sicherheitsexperte Kurt Seifried betreibt eine inoffizielle, moderierte Version von Full Disclosure, auf der er unerwünschte oder überflüssige Postings ausfiltert. Diese Liste können Sie online⁹ bestellen – ob Sie die Moderation durch einen Dritten benötigen, sollten Sie jedoch selber entscheiden.

1.7.3 BugTraq

Im Gegensatz zur ungefilterten Full Disclosure ist BugTraq eine moderierte Mailingliste. Der Moderator David Ahmad überprüft jedes Posting, um Flamewars und »Trolle«, also Störenfriede, nach Möglichkeit fernzuhalten. Der Traffic auf BugTraq ist daher meist um eine Größenordnung geringer als auf FD.

Andererseits sind die Diskussionen dort bei weitem nicht so lebhaft und aufschlussreich wie in der Nachbarliste – meist beschränken sich Postings auf Advisories aller Art. Sicherheitsexperten melden gefundene Lücken in Softwareprodukten, und die Hersteller reagieren mit einer Benachrichtigung, sobald das Problem behoben ist.

Einen Gutteil des Verkehrs auf BugTraq machen Advisories der Anbieter von verschiedensten Linux-Distributionen aus, die ihre Nutzer auf sicherheitsrelevante Updates hinweisen.

BugTraq ist mittlerweile mehr oder minder eine reine Ankündigungsliste, der Diskurs findet inzwischen an anderen Stellen statt. Wenn Sie Zeit sparen möchten, empfiehlt sich ein Abonnement dieser Mailingliste statt eines FD-Abos.

BugTraq können Sie abonnieren, indem Sie eine leere Mail an die Adresse `bugtraq-subscribe@securityfocus.com` senden.

1.7.4 Webappsec

Für Entwickler von Webapplikationen hochinteressant ist die Liste WebAppSec (kurz für »Web Application Security«). Hier tauschen sich Webentwickler jeder Couleur über Sicherheitsfragen aus. Nicht nur Lücken und Exploits gehören zum Thema der Liste, sondern auch Diskussionen über den Einsatz von SSL, Webserver-Sicherheit und Firewalls. Auf der Liste gehen üblicherweise nicht mehr als zehn Postings pro Tag ein, und die Schnittmenge mit Beiträgen auf Bugtraq oder Full Disclosure ist praktisch Null. Daher sollten Sie ein Abonnement in Erwägung ziehen – Postings sind in der Regel von hoher Qualität und für Webentwickler interessant.

8. <https://lists.grok.org.uk/mailman/listinfo/full-disclosure>

9. <http://lists.seifried.org/mailman/listinfo/security>

Auch für ein WebAppSec-Abonnement genügt eine leere Mail, in diesem Fall an die Adresse webappsec-subscribe@securityfocus.com.

1.8 OWASP

Eine der wichtigsten Ressourcen für an Sicherheit interessierte Webentwickler ist das Open Web Application Security Project, kurz OWASP. Hier versammeln sich Programmierer aus aller Herren Länder, um gemeinsame Richtlinien für Web Security zu definieren und zu pflegen. Eine sehr umfangreiche Bibliothek enthält Checklisten, How-Tos und Filter-Bibliotheken für verschiedene Sprachen.

*Open Web Application
Security Project*

Ein Ziel des OWASP ist es, feste Standards zu definieren, die jeder Entwickler befolgen sollte, um ein Mindestmaß an Sicherheit für seine Applikationen garantieren zu können. Das Projekt stützt sich hierbei besonders auf die ISO-Richtlinie 17799 (bzw. ihr britisches Äquivalent BS7799). Zusätzlich gehen die Projektmitglieder auf andere internationale Standards für Sicherheit im Allgemeinen und speziell für Applikationssicherheit ein – für jeden Entwickler, dessen Software auf der ganzen Welt eingesetzt werden soll, ist die Konformität mit diesen Standards unverzichtbar.

Von besonderem Interesse für jeden PHP-Entwickler ist der »Guide to Building Secure Web Applications and Web Services«¹⁰. Dieses über 200-seitige Dokument enthält zahlreiche Anregungen und Best Practices für Entwickler webbasierter Applikationen – auch die Autoren dieses Buches konnten noch das eine oder andere vom OWASP-Guide lernen.

*Guide to Building
Secure Web Applications
and Web Services*

Ein weiteres interessantes Dokument ist die »OWASP Web Application Penetration Checklist«, die als Grundlage für die Security-Checkliste im Anhang diente. Neben den auch in unserem Buch enthaltenen sicherheitsrelevanten Punkten zum Abhaken geht sie auch auf den Ablauf eines »Pentests«, also eines Penetrationstests für Webapplikationen ein, um Entwicklern und externen Sicherheitsexperten Hilfestellung für Sicherheitsüberprüfungen zu geben.

*OWASP Web Application
Penetration Checklist*

Das als PDF erhältliche Dokument steht – ebenso wie der »Guide for Building Secure Web Applications« – unter der GNU Documentation License und kann von jedermann frei verwendet, geändert und für die eigene Dokumentation eingesetzt werden.

Die Teilnahme an der OWASP steht jedem offen – es werden für einige Themen noch Freiwillige gesucht, die Inhalte liefern können.

10. <http://www.owasp.org/documentation/guide.html>

1.9 PHP-Sicherheit.de

Natürlich gibt es zu diesem Buch auch eine Website, getreu dem Motto »Nichts ist so alt wie die Sicherheitslücke von gestern«. Neben Errata und Aktualisierungen werden wir versuchen, in einem Weblog aktuelle Lücken in PHP-Applikationen aufzulisten und zu kommentieren. So können sich PHP-Entwickler an einem Ort über Lücken und Bugs in der von ihnen eingesetzten Software informieren und sparen sich im besten Falle das Abonnement der oben aufgeführten Security-Mailinglisten.

Darüber hinaus werden Artikel zu PHP-Sicherheit und Vorträge über dieses Thema auf der Website¹¹ zum Buch veröffentlicht.

11. <http://www.php-sicherheit.de/>