Einleitung

Sicherer Programmcode

Dieses Buch ist irgendwie sonderbar. Es handelt von der Sicherheit einer Website, erwähnt aber kaum Firewalls. Es handelt von der Sicherheit von Informationen, sagt aber sehr wenig über Verschlüsselung. Was hat es also mit diesem Buch auf sich? Es beschreibt einen kleinen, oft vernachlässigten Aspekt der Website-Sicherheit: sicheren Programmcode.

Firewall, Verschlüsselung, Software-Patches

Viele glauben, dass eine gute Firewall, verschlüsselte Kommunikation und die jeweils neuesten Software-Patches alles sind, was man braucht, damit eine Website sicher ist. Sie irren sich. Viele der heutigen Websites enthalten Programmcode, der ihre Webseiten dynamisch erzeugt. Dazu gehört Code, der mit Tools wie Java, PHP, Perl, ASP/VB-Script, Zope, ColdFusion usw. entwickelt wurde. Viel zu oft wird dieser Code von Programmierern geschrieben, die anscheinend glauben, dass Sicherheit Sache der Administratoren ist. Das führt dazu, dass viele dynamische Websites logische Löcher aufweisen, durch die sie allen Arten gemeiner Attacken ausgesetzt sind – selbst mit vorhandener Firewall und Verschlüsselung.

Die derzeitige Ausbildung von Programmierern betrachtet Sicherheit meist als Randthema, das die Administratoren oder eine kleine Elite von Sicherheitsspezialisten betrifft. Wir lernen, wie man programmiert. Punkt. Genauer gesagt, wie man Programme macht, die den Kunden gefallen, indem die geforderte Funktionalität umgesetzt wird. Vor einigen Jahren hat das wahrscheinlich genügt. Damals waren Programme eine unternehmensinterne Angelegenheit. Jede Person mit Zugriff auf unser Programm erwartete, dass es korrekt funktionierte und dass sie damit ihre tägliche Arbeit erledigen konnte.

Programme für das Web

Im Zeitalter des Web wird vermehrt von uns gefordert, Programme zu erstellen, die der ganzen Welt zur Verfügung stehen. Legitime Benutzer wollen nach wie vor, dass das Programm einfach jene Arbeiten erledigt, für die es vorgesehen ist. Leider steht unser Programm auch vielen Leuten zur Verfügung, die Spaß daran haben, Programme zu knacken oder – noch schlimmer – sie Dinge machen zu lassen, für die sie nicht erdacht worden sind.

Exploits

Bis vor kurzem haben solche Leute, den größten Teil ihrer Bemühungen in massenproduzierte Software gesteckt und so genannte Exploits (Hacking-Werkzeuge) ersonnen, die auf Tausenden von Sys-

temen funktionieren. In den letzten paar Jahren wenden sie sich jedoch verstärkt individuell zugeschnittenen Web-Anwendungen zu. Gleichzeitig wurden internationale Sicherheits-Mailinglisten geschaffen, die sich nur mit der Web-Anwendungsschicht befassen, zahlreiche White-Papers wurden geschrieben, und in den Medien lesen wir von den ersten Attacken auf Anwendungsebene. Vor dem Hintergrund dieser Schwerpunktverlagerung werden aller Wahrscheinlichkeit nach immer mehr Angreifer dazu übergehen, mit Anwendungs-Exploits zu arbeiten. Während sich die Sicherheitsleute bemühen, damit Schritt zu halten, hinken die Programmierer weit hinterher. Es ist an der Zeit, dass *wir* beginnen, uns auch mit Sicherheit zu befassen.

Die Leser

Dieses Buch wurde für diejenigen unter uns geschrieben, die dynamische Web-Anwendungen programmieren. Das Buch erklärt viele häufige Fehler, die diese Programmierer machen, und wie diese Fehler zum Vorteil der Angreifer ausgenutzt werden können.

Denken wie ein Angreifer

Während Sie das Buch lesen, bekommen Sie vielleicht den Eindruck, dass der Schwerpunkt darauf liegt, wie eine Website geknackt wird – statt zu lernen, wie man eine Site baut, die dagegen geschützt ist. Der Schwerpunkt ist absichtlich so gewählt: Will man sichere Anwendungen entwickeln, muss man wissen, wie Programmierfehler missbraucht werden können. Man muss wissen, wie der Angreifer denkt, wenn er auf der Suche nach Schwachstellen herumschnüffelt. Um unseren Code zu schützen, müssen wir den Feind kennen. Die beste Art, einen Angreifer aufzuhalten, besteht darin, wie ein solcher zu denken.

Dieses Buch hat nicht zum Ziel, Ihnen alles darüber zu sagen, wie man sichere Web-Anwendungen schreibt. Ein solches Buch würde Tausende von Seiten umfassen und wäre ziemlich langweilig: Es würde haufenweise Details über jede in der Praxis vorhandene Web-Programmiersprache enthalten, von denen Sie die meisten ohnehin nie benutzen. Und es würde jede Menge Details über Probleme enthalten, um die Sie sich nie kümmern werden. Jede Programmierplattform und jede Problemart haben ihre eigenen Fallstricke.

Das Ziel des Buchs

Das Ziel dieses Buchs ist es, Sie darauf aufmerksam zu machen, dass der von Ihnen geschriebene Code ausgenutzt werden kann, und dass es ungeachtet der von Ihnen benutzten Plattform viele Fallen gibt. Hoffentlich wirkt dieses Buch auf Sie wie ein Quälgeist oder Weckruf, durch den Sie erkennen, dass die Programmierung, mit der Sie Ihren

Lebensunterhalt verdienen, tatsächlich ein wichtiger Teil der gesamten Sicherheitslandschaft ist. Wenn Sie nach der Lektüre dieses Buchs beim Programmieren ein bisschen paranoider geworden sind, hat dieses Buch sein Ziel erreicht.

Die Regeln

Best Practices

Im Verlauf der Kapitel dieses Buchs werden Sie auf eine Hand voll »Regeln« oder »Best Practices« stoßen. Diese Regeln heben Punkte hervor, die man verstehen und sich merken sollte. Wie die meisten Regeln sind auch diese nicht ohne Ausnahmen. Einige davon können zurechtgebogen, andere gebrochen werden. Bevor Sie sich aber anschicken, eine Regel zurechtzubiegen oder zu brechen, sollten Sie das Sicherheitsproblem, das die Regel zu verhindern versucht, sehr genau verstanden haben. Wenn Sie die Regel absichtlich nicht befolgen, sollten Sie sehr gut einschätzen können, warum Ihre Anwendung trotzdem sicher bleibt oder warum es eventuell keine Rolle spielt, wenn sie denn verwundbar ist.

Die Feststellung, ob eine Anwendung nicht verwundbar ist, ist nicht unbedingt eine einfache Aufgabe. Es ist leicht sich zu sagen: »Wenn ich in meinem Code keine Sicherheitslöcher finde, wird schon niemand anders welche finden.« Es ist aber auch extrem gefährlich. Der Durchschnittsentwickler ist nicht im destruktiven Denken geschult. Er oder sie arbeitet, um Dinge zu konstruieren. Es mag immer einen spitzfindigen Eindringling geben, wenn es darum geht, böswilliger zu denken als der Entwickler. Um sich das vor Augen zu führen und gleichzeitig zu sehen, wie die Regeln aufgebaut sind, führen wir die erste Regel ein:

Regel 1
Unterschätze nicht die Macht der dunklen Seite.

Nie im Schnellverfahren arbeiten

Diese Regel fordert uns auf, nie im Schnellverfahren zu arbeiten oder gar bekannte Sicherheitsmaßnahmen zu ignorieren – ganz unabhängig davon, welches Programm wir erstellen und an welchem Teil des Programms wir momentan arbeiten. Sie ermutigt uns auch, etwas paranoid zu sein. Die Regel wirkt an sich nicht besonders überzeugend. In Verbindung mit dem Inhalt dieses Buchs sollte sie es aber hoffentlich sein. Das Web hat eine finstere Seite. Da draußen gibt es einen, der nach einer Gelegenheit sucht, eine Website entweder zum Spaß oder zum Profit zu hacken. Solche Leute können die Website, an deren Erstellung Sie Monate gearbeitet haben, ruinieren, ganz egal, welche

Absichten dahinter stecken. Doch auch wenn kein direkter Schaden angerichtet wird, können die Auswirkungen von Sicherheitslücken sowohl für die Website als auch für die Firma, die sie erstellt hat, zu sehr schlechter Presse führen.

Die Beispiele

Dieses Buch enthält sehr viele Beispiele. Der Autor meint, dass Beispiele und praktisches Experimentieren die beste Art sind, etwas zu lernen. In Zusammenhang mit Sicherheit lassen sich die beiden Lernmechanismen aber nicht immer kombinieren. Verwenden Sie die Beispiele in diesem Buch *nicht* zum Experimentieren auf Sites, ohne vorher die ausdrückliche Erlaubnis der Site-Eigentümer einzuholen. Je nach den Gesetzen Ihres Landes könnten Sie sonst im Gefängnis landen.

Aus der wirklichen Welt

Viele der Beispiele erwecken den Eindruck, als würden sie Anwendungen aus der wirklichen Welt beschreiben. Und genau das tun sie auch. Die echt anmutenden Beispiele basieren auf Code-Reviews und Tests durch verschiedene Personen, einschließlich des Autors. Einige Beispiele beruhen sogar auf nicht autorisierten, aber unschädlichen Experimenten (zum Glück sitze ich noch nicht im Gefängnis). Ich habe die Sites anonymisiert, indem ich ihre Namen nicht nenne und oft auch Codeteile in einer anderen als der für die Site tatsächlich benutzten Programmiersprache darstelle.

Code in Java, PHP, Perl und VBScript

Die Beispiele sind im Wesentlichen kleine, in Java, PHP, Perl oder VBScript geschriebene Codeauszüge. Diese Sprachen dürften für die meisten Programmierer recht leicht zu lesen sein. Wenn eine dieser Sprachen für Sie neu ist, finden Sie die folgende Tabelle wahrscheinlich hilfreich; sie listet ein paar syntaktische Unterschiede auf:

	Java	PHP	Perl	VBScript
Zeichenverkettung	+	.	.	&
Variablenpräfix		$	$	
Subroutinenpräfix			&	
Zeilenfortführung				_

Die in den Beispielen benutzten Domainnamen folgen den im RFC[1] 2606 [1] definierten Richtlinien. Keiner der Domainnamen ist in der wirklichen Welt gültig. Die IP-Adressen sind private Adressen gemäß RFC 1918 [2] und im Internet nicht gültig.

Für eine bessere Lesbarkeit wurden einige Beispieltexte im Layout umbrochen. Lange URLs, Fehlermeldungen und Textketten, die nor-

malerweise in einer Zeile stehen, aber das Buchlayout überschreiten würden, können sich daher im Folgenden über mehrere Zeilen erstrecken, ohne dass ein besonderer Hinweis erfolgt.

Die Kapitel

Obwohl dieses Buch so geschrieben wurde, als ob der gesamte Text sequenziell gelesen würde, können Sie natürlich auch einzelne Kapitel lesen. Hier eine Kapitelübersicht:

- Kapitel 1 enthält eine Einführung in HTTP und damit zusammenhängende Web-Technologien (wie Cookies und Sessions) sowie Beispiele, die zeigen, was schief laufen kann, wenn wir nicht verstehen, wie alles funktioniert.
- Kapitel 2 befasst sich mit Metazeichenproblemen, die auftauchen können, wenn wir Daten an ein anderes System übergeben. In diesem Kapitel wird das berühmte SQL-Injection-Problem ausführlich behandelt.
- Kapitel 3 beschreibt die Eingabebehandlung, also wie man ungültige Eingaben erkennt, wie man sie behandelt und warum man nicht blind dem vertrauen sollte, was vom Client kommt.
- Kapitel 4 zeigt, wie wir uns großen Ärger einhandeln können, wenn wir Daten an die Browser unserer Benutzer senden, ohne sie vorher zu filtern. In diesem Kapitel wird das Cross-Site-Scripting-Problem beschrieben.
- Kapitel 5 erklärt, wie leicht man einen Benutzer dazu bringen kann, etwas im Web zu tun, was er nie beabsichtigt hat, und zwar einfach dadurch, dass man ihn auf eine spezielle Webseite lockt oder ihm eine E-Mail sendet.
- Kapitel 6 befasst sich mit Passwortbehandlung, geheimen Kennungen und anderen Dingen, die wir vor Eindringlingen verstecken wollen. Dieses Kapitel beinhaltet die wohl weltweit kürzeste Einführung in die Kryptologie.
- Kapitel 7 diskutiert Gründe, warum der Code von Web-Anwendungen bei seiner Fertigstellung oft noch Sicherheitslücken aufweist.
- Kapitel 8 führt alle Regeln der vorherigen Kapitel auf und enthält kurze Zusammenfassungen.

1. RFC ist die Abkürzung für *Request For Comments*; RFCs enthalten technische und organisatorische Dokumente über das Internet, die vom RFC-Editor [3] für die Internet Engineering Task Force IETF [4] gepflegt werden. Jedes offizielle Internetprotokoll ist in einem oder mehreren RFCs definiert.

- Schließlich folgen Anhänge über Webserver-Fehler, Packet-Sniffing, E-Mail-Fälschung und Quellenhinweise zu weiteren Informationen. Notorische Anhangüberspringer sollten zumindest den Teil »Weitere Informationen« lesen.

Das Buch enthält außerdem ein Literaturverzeichnis. Durch das gesamte Buch hindurch begegnen Sie Zahlen zwischen [eckigen Klammern]. Diese Zahlen beziehen sich auf die Einträge im Literaturverzeichnis. Die Einträge verweisen auf Bücher, Artikel und Websites mit weiteren Informationen zu den jeweils behandelten Themen.

Was ist nicht in diesem Buch?

Infrastruktur und Sicherheitsdesign

Da dieses Buch für Programmierer geschrieben ist, wurden viele Sicherheitsmaßnahmen zur Infrastruktur außen vor gelassen. Außerdem fehlt Sicherheitsdesign fast vollständig, d. h. welche Authentifizierungsmethoden wir verwenden, wie Logik in mehrere Schichten auf mehrere Server aufgeteilt werden kann usw. Beim Schreiben des Programmcodes wurden diese Entscheidungen bereits getroffen. Hoffentlich! Wenn Sie nicht nur programmieren, sondern auch Sicherheitsmechanismen entwerfen, empfehle ich Ihnen unbedingt, *Security Engineering* [5] von Ross Anderson zu lesen. In diesem Buch wird aufgezeigt, wie leicht es ist, etwas falsch zu machen (und wie man's nicht macht).

Buffer Overflows

Ein wichtiges Thema, das auf der Liste von C/C++-Programmierern ganz oben stehen sollte, wurde ebenfalls weggelassen: das Buffer-Overflow-Problem. Dieses Problem ist für alle, die in C/C++-Programmierung keine alten Hasen sind, sehr schwer zu verstehen. Wenn Sie in C, C++ oder einer anderen Sprache programmieren, die keine Pointer- und Indexkontrollen usw. hat, dann sollten Sie auf jeden Fall verstanden haben, wie wichtig es ist, Ihre Speicherbereiche zu schützen. Ich schlage vor, dass Sie einen Blick in Aleph Ones klassischen Artikel »Smashing the Stack for Fun and Profit« [6] werfen oder sich ein allgemeines Buch über sichere Programmierung besorgen, das diese Dinge ausführlich erklärt. Ich empfehle *Building Secure Software* [7] von John Viega und Gary McGraw.[2]

Da wir schon beim Thema Bücher über sichere Programmierung sind, sollte ich auch *Writing Secure Code* [8] von Michael Howard und David LeBlanc sowie die Onlinearbeit »Secure Programming for Linux

2. In Deutsch existiert zu diesem Thema das Buch »Buffer Overflows und Format-String-Schwachstellen« von Tobias Klein, dpunkt.verlag, 2003. (Anm. d. Übers.)

and Unix HOWTO« [9] von David Wheeler erwähnen. Ersteres hat zwar einen Hang zur Microsoft-Plattform, während die zweite Arbeit Unix und Linux bevorzugt, doch enthalten beide umfangreiche relevante Teile, gleichgültig, was Ihre Plattform ist.

Dieses Buch konzentriert sich auf die serverseitige Programmierung. Es behandelt nicht Java-Applets, ActiveX-Objekte oder andere Technologien, durch die Programme auf der Clientseite laufen können. Wenn Sie clientseitige Programme erstellen, sollte Ihnen klar sein, dass das Programm unter der vollen Kontrolle desjenigen läuft, der den Computer bedient. Außerdem ist es sicherlich angebracht, eines der Bücher über allgemeine Codesicherheit zu lesen.

Plattformabhängige Sicherheitskniffe

Und schließlich wurden die meisten plattformabhängigen Sicherheitskniffe weggelassen, damit das gesamte Buch für jeden lesbar bleibt. Nach der Lektüre dieses Buchs bitte ich Sie eindringlich, einige Zeit dafür aufzuwenden, im Web nach »Sicherheit, Best Practices« für die Plattform Ihrer Wahl zu suchen.

Anmerkung des Autors

Vielleicht interessiert es Sie, dass ich selbst Web-Programmierer bin. Ich habe meinen (nicht geringen) Beitrag zu Sicherheitslücken geleistet. Obwohl ich mich in den letzten drei Jahren jeden einzelnen Tag auf solche Schwachstellen und ihre Vermeidung konzentriert habe, produziere ich sie immer noch. Nun gut, ich bilde mir ein, dass mir heute weniger Fehler unterlaufen als früher. Nicht etwa, dass ich ein besserer Programmierer geworden wäre, sondern weil ich erkannt habe, dass jede einzelne Zeile, die ich schreibe, in puncto Sicherheit zählt, und – vor allem – dass es viel zu leicht ist, Fehler zu machen.

Leserkommentare

Falls dieses Buch Sie ärgert, erfreut, neugierig macht, erschreckt, aufregt, beruhigt oder sonstige Wirkungen hervorruft, teilen Sie mir dies bitte mit und senden Sie mir eine E-Mail (in Englisch) an `innocentcode@thathost.com`. Falls Sie Fehler finden, melden Sie diese bitte an die gleiche Adresse. Sollten Sie sich zufällig in Oslo (der Hauptstadt von Norwegen) aufhalten und die Themen des Buchs bei einem Glas Bier[3] diskutieren wollen, dürfen Sie mich gerne einladen :–)

3. Ich muss Sie warnen: Bier ist in Norwegen recht teuer.

Website zum Buch

Zu diesem Buch gibt es außerdem eine englische Website unter der Adresse `http://innocentcode.thathost.com`. Korrekturen oder Ergänzungen zu diesem Buch werden auf dieser Site erscheinen.