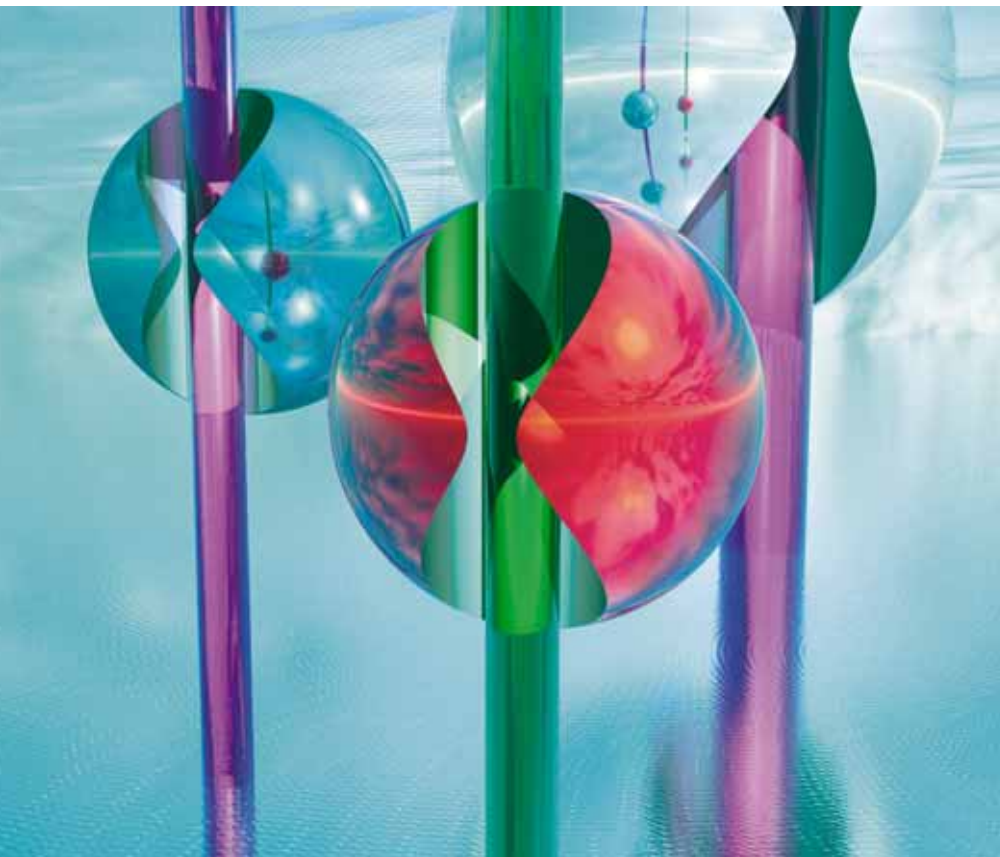


Web Application Security

Teil 1

Die Grund-
problematik

Egal ob Online-Bank, großer Online-Händler oder kleiner Online-Shop - die entscheidende Schnittstelle zwischen Internet und vertraulicher Unternehmensinformation bilden Web-Anwendungen, die die Kommunikation zwischen Kunde und Backend-Systemen „vermitteln“. Aus Hackersicht bieten sie deshalb ein lukratives – und zudem heute oft noch ungeschütztes – Einfallstor direkt zum Ziel – den vertraulichen Unternehmensdaten.

Dieser Artikel gibt – als Auftakt einer Artikelreihe – eine Einführung in das Thema Web Application Security. In den folgenden Artikeln werden jeweils spezielle Angriffsformen und deren Abwehr dargestellt. „Nicht Neugierde und technisches Wetteifern sind die Motive für die gegenwärtigen Angriffsaktivitäten, sondern finanzieller Profit durch den Diebstahl vertraulicher Daten, Erpressung, Betrug oder ganz einfach die Bereitstellung von Hackerdienstleistungen“ (Quelle: Quintessenz des Symantec Internet Security Threat Report, 1. Halbjahr 2005).

Diese Motivationslage von Hackern vorausgesetzt ist es keine Überraschung, dass gezielte Angriffe auf Webanwendungen und Webbrowser – neben BotNets und anpassbaren modularen Schadcodes – laut Symantec zu den wichtigsten Ausprägungen des aktuellen Bedrohungsszenarios zählen. Um zu verstehen, weshalb Angriffe auf Webanwendungen heute auch sehr oft erfolgreich sind, lohnt es sich, folgende zwei Kernfragen genauer zu betrachten: Wieso bieten bereits vorhandene traditionelle IT-Sicherheitssysteme wie Firewalls oder Intrusion Detection/Prevention-Systeme keinen ausreichenden Schutz gegen derartige Angriffe? Welche zusätzlichen Angriffspunkte weisen Web-Applikationen im Vergleich zu klassischen, lokal installierten Anwendungen auf?

Anwendungsschicht vs. Transportschichten

Zur Beantwortung der ersten Frage hilft ein Blick in die technischen Grundlagen der Kommunikation im Internet. Diese lässt sich gut darstellen anhand des ISO/OSI-7-Schichten-Modells (siehe Bild 2).

Die Daten durchlaufen alle Schichten und werden jeweils in schichtenspezifische Darstellungen umgewandelt. Die höchste Schicht entspricht der Applikationsebene; dort werden die Daten verarbeitet, die darunter liegenden Schichten dienen lediglich dem Transport von

Applikation zu Server (und umgekehrt). Auf diese Weise werden Daten von der Applikation auf dem Web-Server zur Applikation auf dem Client, dem Web-Browser, kommuniziert. Jetzt kommt der entscheidende Punkt: Vor zehn Jahren waren alle Schichten ungeschützt – Angriffe auf den unteren Ebenen des Sieben-Schichtenmodells waren einfach und hatten große Wirkung. Als Reaktion gegen Angriffe auf diesen Ebenen entstanden die heute üblichen IT-Sicherheitslösungen wie Firewalls und Intrusion Detection-Systeme. Mit zunehmender Absicherung der Transportschichten einerseits – und aus den oben genannten Motivationsgründen andererseits – wurde die Applikationsebene selbst für die Angreifer immer lukrativer.

Die genannten herkömmlichen IT-Sicherheitslösungen können – letztlich auch historisch bedingt – einen HTTP-Request nicht weiter prüfen; würden sie jeden HTTP-Request blocken, gäbe es überhaupt keine Kommunikation. Also müssen sie jeden HTTP-Request durchlassen! Mit anderen Worten: Die bekannten klassischen IT-Sicherheitssysteme wurden zum Schutz der Kommunikation auf Transportebene entwickelt – und hier leisten sie auch gute Arbeit. Zum Schutz auf Anwendungsebene dienen sie aber nur in sehr geringem, nicht ausreichendem Maß. Hinzu kommt, dass die Vielfalt der angebotenen Web-Script-Sprachen, Application Frameworks und Webtechnologien eine fast unbegrenzte Anzahl von Sicherheitslücken erzeugt – eine ideale Ausgangsposition für Hacker.

Typische Schwachstellen von Webapplikationen

Im folgenden wenden wir uns der Beantwortung der zweiten Kernfrage nach den typischen zusätzlichen Angriffspunkten von Webapplikationen zu. Diese resultieren im Kern aus der netzwerkbasieren Natur von Webapplikationen. Im folgenden sollen die größten Problemfelder aufgezeigt werden.

Die meisten Programmierer und ihre Programme haben ein sehr menschliches Problem: Sie vertrauen dem Benutzer der Applikation.

Überprüfung aller Eingaben

Das daraus resultierende klassische Problem hierbei ist der Buffer Overflow: Eine Applikation erwartet ein Wort oder eine Zeile als Eingabe des Benutzers und ist nicht darauf vorbereitet, dass der Benutzer mehr als 1024 Zeichen oder gar 2 GigaByte an Daten eingibt. Dann werden von der Eingabe plötzlich andere Teile des Programms überschrieben. Im Bereich der Webapplikationen spielt der einfache Buffer-Overflow keine so grosse Rolle mehr, da sie meistens

Angriffers) ungeprüft an andere Komponenten des Gesamtsystems weitergeleitet werden. Hieraus resultiert dann die Problemklasse der Command-Injection-Angriffe; am bekanntesten dürfte hier die SQL-Injection sein. Hierbei kann ein Angreifer direkt Kommandos auf der zur Webapplikation üblicherweise gehörenden Datenbank ausführen, fremde Daten auslesen oder manipulieren. Etwas älter sind die Angriffe auf das Betriebssystem des Webservers.

Erschwerend kommt hinzu, dass es keineswegs offensichtlich ist, welche Daten tatsächlich Benutzereingaben sein können. Oft wird fälschlicherweise angenommen, dass von der Applikation gesetzte Cookies, ver-

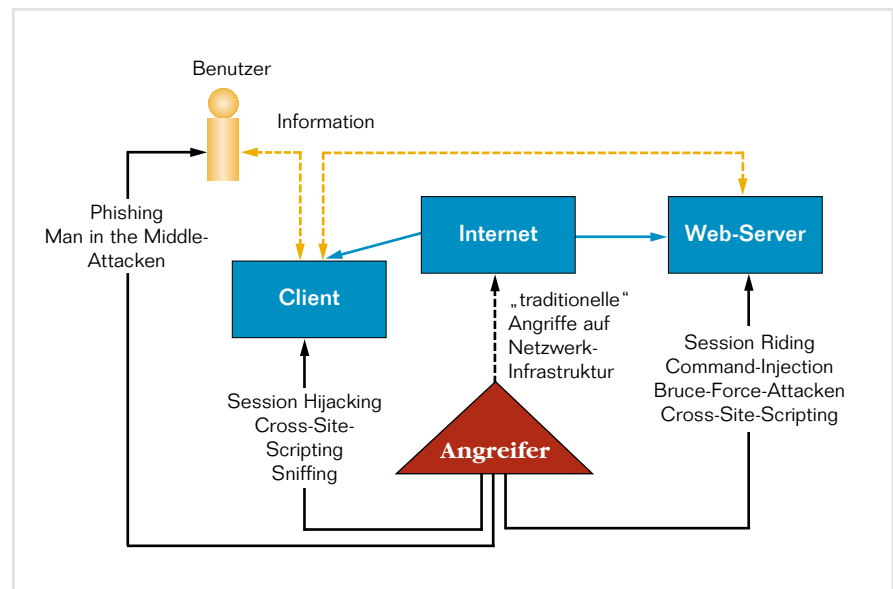


Bild 1: Verschiedene Angriffsmethoden auf Webanwendungen im Überblick.

in Java, PHP, Perl oder anderen Sprachen mit automatischer Speicherverwaltung geschrieben werden. Lediglich der Webserver selbst – und hier in der Vergangenheit häufig die SSL-Bibliothek – ist für solche Angriffe noch anfällig.

Eine Ausnahme hiervon sind natürlich Applikationsserver, die in C oder C++ geschrieben sind. Ein prominentes Beispiel hierfür mit Problemen in der Vergangenheit ist das SAP Web-Frontend. Dafür treten immer dann Probleme auf, wenn Eingabedaten des Benutzers (oder des

steckte Eingabefelder oder zum Beispiel die Namen von Auswahlboxen in einem Webformular nur unverändert wieder als Eingabe erscheinen können. Tatsächlich sind aber alle genannten Beispiele von einem Angreifer frei manipulierbar. Selbst der Hostname des Clients kann unter Umständen für Angriffe benutzt werden.

Session Handling

Prinzipiell ist das HTTP Protokoll zustandslos. Zu Zeiten von HTTP Version 1.0 lief eine Anfrage an einen Webserver wie folgt ab: Der Web-

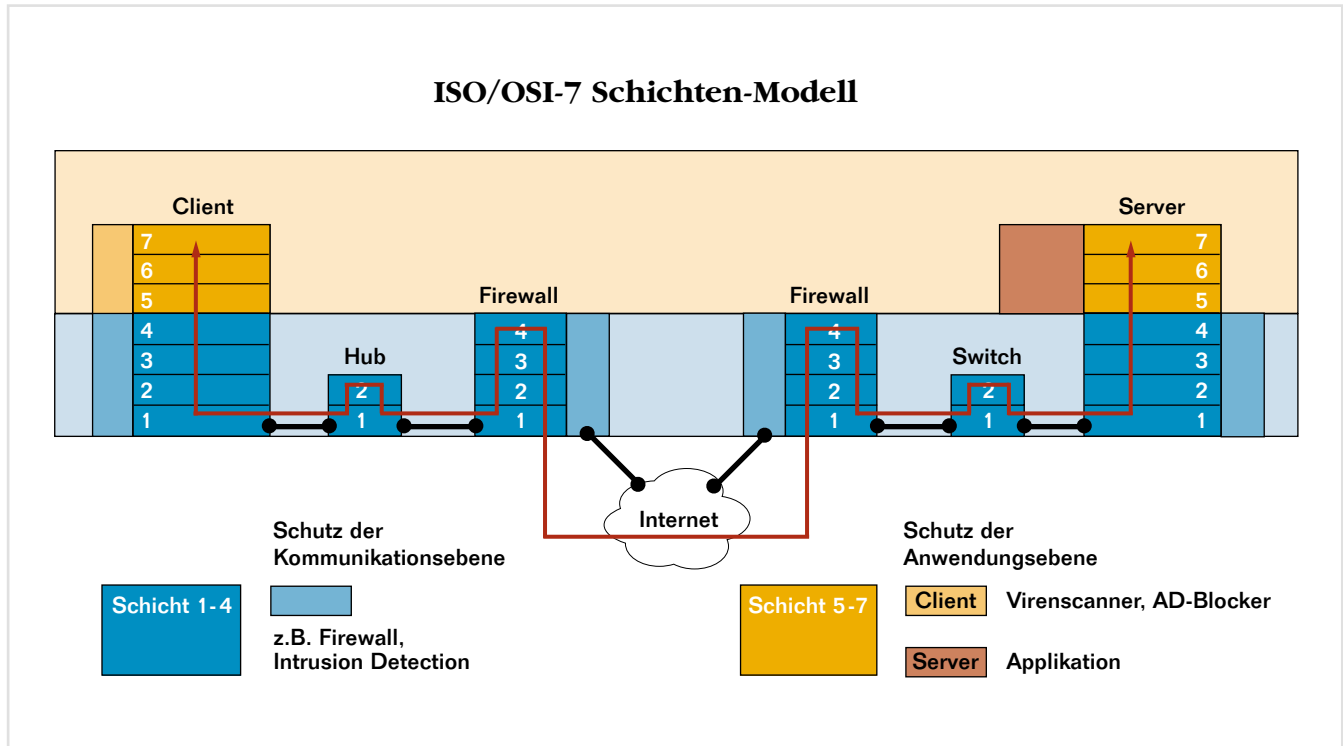


Bild 2: Kommunikation im Internet anhand 7-Schichten-Modell

browser stellt eine Verbindung zum Server auf Netzwerkebene her und sendet dem Server die URL der angeforderten Webseite. Der Server schickt dann ein Dokument zurück und die Verbindung ist beendet. Die nächste Webseite oder auch schon die in einer Webseite eingebetteten Bilder werden über eine neue Verbindung angefordert. Im wesentlichen hat sich daran auch im aktuellen Protokoll HTTP 1.1 nichts geändert. Zwar wird die Netzwerkverbindung nicht mehr für jeden Request neu aufgebaut, sondern mehrere Webseiten werden über dieselbe Netzwerkverbindung geladen.

Dies ist allerdings nur eine Performance-Optimierung, vom Standpunkt der Applikation aus besteht die Kommunikation immer noch aus

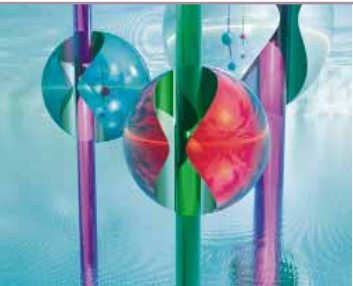
einzelnen unabhängigen Anfragen. Das Web war ja ursprünglich nur als Medium gedacht, um Inhalte zu publizieren. Die Idee, damit komplexere Applikationen wie einen Online-Shop zu realisieren, kam erst später auf. Ein Online-Shop muß zusammengehörende http-Anfragen auch als zusammengehörig (zu einer Session gehörend) erkennen. Dafür müssen Informationen von einer Anfrage an die nächste übergeben werden.

Oftmals sind diese Sessions nicht besonders geschützt, das heisst ein Angreifer kann eine bestehende Session eines anderen Benutzers übernehmen (Session Hijacking). Da an diese Session in der Regel Authentisierungsinformationen gebunden sind, kann der Angreifer dann im Namen des fremden Nutzers agieren,

zum Beispiel auf seine Rechnung Einkäufe tätigen.

State

Fast jede Applikation heutzutage ist stateful (zustandsbehaftet), dass heisst, es gibt einen inneren Zustand der Applikation, der im Verlauf der Benutzung aktualisiert wird und von dem das weitere Verhalten der Applikation abhängig ist. Ein einfaches Beispiel hierfür ist der Warenkorb beim Online-Shopping. Während bei klassischen Programmen dieser Zustand implizit durch das Speicherabbild des Programms zur Laufzeit gegeben ist, muss dieser Zustand bei Webapplikationen explizit abgelegt werden. In der Regel werden dafür Cookies oder die Parameter in der URL benutzt. Diese können jedoch vom Benutzer manipuliert werden. Der Applikationsentwickler muss also eine klare Entscheidung zwischen sicherheitsrelevanten und unkritischen Parametern treffen. Diese Entscheidung ist nicht immer einfach und eindeutig zu treffen. Deshalb ist es manchmal möglich, dass ein Angreifer den Warenkorb eines Webshops editieren und die Preise der



„Typische Angriffspunkte resultieren im Kern aus der netzwerkbasiereten Natur von Webapplikationen“

Artikel ändern kann. Wenn dann noch der gesamte Bestellvorgang automatisiert ist und keine Plausibilitätsprüfung durch einen Menschen erfolgt, hat der Betreiber des Webshops ein ernstes Problem.

Angreifbarkeit aus dem Netz

Klassische Applikationen sind nur angreifbar, wenn der Angreifer Zugriff auf den Rechner oder das lokale Netzwerk hat. Dafür gibt es erprobte Schutzmittel wie Türschlösser, User Accounts, Firewalls und Intrusion-Detection-Systeme. Leider nutzen diese Techniken im Web Applikations-Bereich nur wenig. Die Applikation soll ja gerade aus der ganzen Welt zugreifbar sein. Das impliziert, dass sie auch aus der ganzen Welt angreifbar ist. Viele Webapplikationen setzen heute auf Standardsoftwarepaketen auf. Ist in diesen erst einmal ein Fehler gefunden, können in kurzer Zeit genau diese Fehler automatisch in allen Webapplikationen ausgenutzt werden. Nahezu jede Website wird heute von Angreifern regelmäßig nach bekannten Schwachstellen oder Konfigurationsfehlern des Admins gescannt. Ein Blick in die Server-Logfiles lohnt sich immer...

Ein weiteres Problem mit Netzwerkanwendungen generell ist die Anfälligkeit der Verbindung zwischen Client und Server, hier also zwischen dem Webbrowser des Benutzers und dem Webserver, welcher die Applikation ausführt. Wenn nicht besondere Schutzmechanismen wie SSL benutzt werden, liegt die Kommunikation zwischen Benutzer und Server im Klartext vor, kann also von jedem Provider auf dem Weg zwischen dem Benutzer und dem Webserver mitgelesen und auch modifiziert werden. Nun könnte man einwenden, dass Internetprovider ein gewisses Vertrauen genießen und die Daten schon nicht ändern werden. Aber was, wenn sich ein böswilliger Angreifer in die Kommunikation einklinken kann. Internetverbindungen sind keine Punkt zu Punkt Verbindungen zwischen

Info

Viele Web-Anwendungen benutzen irgend eine Form von Session-Management, um eine an den Benutzer angepasste Umgebung zu schaffen. Die mit der Session-ID verknüpfte Information ist ein attraktives Ziel für Angreifer. Der zweite Artikel unserer Serie behandelt Angriffstechniken wie Session-Prediction – Interception – Fixation oder Brute-Force-Attacken und deren Abwehr, bei der die Themen Authentisierung und Autorisierung die Hauptrolle spielen.

dem Client und dem Server. Die Datenpakete einer Verbindung können umgeleitet werden - und ein geschickter Angreifer kann diese über seinen Rechner laufen lassen. Das sind dann die sogenannten „Man-In-The-Middle-Attacken“, da der Angreifer zwischen den beiden berechtigten Parteien sitzt. Technische Lösungen wie Verschlüsselung und Authentisierung mit SSL können hier helfen, werden aber oftmals falsch eingesetzt.

Authentizität der Applikation

Lokal auf dem Computer installierte Applikationen haben einen großen Vorteil: Der Benutzer weiss in der Regel genau, in welche Applikation er gerade seine Daten eingibt. Der Nutzer (beziehungsweise der Administrator) hat Kontrolle über die lokal installierten Applikationen, diese sind prinzipiell vom Standpunkt des Benutzers aus vertrauenswürdig. Lokale Applikationen speichern sensible Daten auf der lokalen Festplatte des Benutzers beziehungsweise auf dem vertrauenswürdigen Fileserver im Firmennetz ab. Zugegeben, mit der heutigen Verbreitung von Viren und Trojanern ist diese ideale Welt auch bedroht, aber im Grossen und Ganzen stimmt das Bild noch. Im Webbereich wird es deutlich komplizierter. Der Benutzer kommuniziert mit einer fremden Applikation irgendwo im Netzwerk. Er hat keine Ahnung, wo seine Daten

gelagert werden oder ob sie sicher sind. Jedes Jahr schafft es mindestens ein Skandal über gestohlene Kreditkartendaten in die Medien, zuletzt waren es 40 Millionen Kreditkartendaten.

Aber es kommt noch schlimmer: Selbst wenn der Benutzer der Webapplikation beziehungsweise deren Betreiber in Fragen der Datensicherheit vertraut (die meisten Menschen würden zum Beispiel ihrer Bank vertrauen) - im Web Bereich kann ein Benutzer nicht unbedingt erkennen, ob er gerade mit der „richtigen Applikation“ spricht oder ob er seine vertraulichen Informationen einer Webapplikation übergibt, die sich nur den Anschein gibt, legitim zu sein. Dieses Problem ist unter den Namen Phishing und Pharming in den letzten Jahren mit zu einem Hauptproblem im Web Application Security-Bereich geworden. Auch hier gibt es technische Lösungen wie SSL Zertifikate; diese haben sich in der Praxis aber als leider nicht ausreichend herausgestellt.

Kurzzusammenfassung und Ausblick

Angriffe auf Webanwendungen haben sich in den letzten Monaten zu einem Kernthema im Bereich IT-Sicherheit entwickelt. Herkömmliche IT-Sicherheitslösungen wie Firewalls bieten keinen ausreichenden Schutz gegen derartige Angriffe, da ihr Hauptziel der Schutz der Transportschichten ist. Typische Schwachstellen von Webanwendungen wurden in diesem Artikel im Überblick dargestellt. Deren Wirkungsweise sowie die Möglichkeiten zur gezielten Abwehr werden in den nachfolgenden Ausgaben im Detail aufgezeigt.

alexander.meisel@artofdefence.com



Alexander Meisel arbeitet seit mehreren Jahren auf dem Gebiet der Web-Security und ist Geschäftsführer und ein Mitbegründer der art of defence GmbH, Regensburg.

