Die größten Schwachstellen in Web-Anwendungen

Interaktiv konzipierte Web-Applikationen sind für Hacker ein gefundenes Fressen – vor allem wenn sie Schwachstellen aufweisen. Lesen Sie, wo die gefährlichsten Lücken klaffen.

Von **Christoph Wolfert***

Über die Jahre hat sich die statische Web-Seite zu einer komplexen dynamischen Anwendung gewandelt, die dem Internet-Nutzer viele Funktionen bereitstellt. Dazu ist eine Vielzahl von Technologien erforderlich, deren Implementierung und Betrieb nicht immer trivial sind. Parallel zum technischen Fortschritt sind neue Schwachstellen in den Anwendungen entstanden. Nach einer Statistik des Web Application Security Consortium lassen sich mittels Penetrationstests in 96,85 Prozent aller Web-Anwendungen kritische Sicherheitslücken identifizieren.

Um Qualität und Sicherheit von Anwendungen zu verbessern, wurde das Community-Projekt „Open Web Application Security Project“ (Owasp) ins Leben gerufen. Innerhalb des Owasp gibt es verschiedene Teilprojekte, so etwa das „Owasp Top Ten Project“, das regelmäßig die jeweils zehn kritischsten Schwachstellen von Web-Applikationen beschreibt (letzter Stand: 2007). Ziel ist es, Entwickler, Designer, Architekten und Unternehmen für potenzielle Schwachstellen zu sensibilisieren und aufzuzeigen, wie sich diese vermeiden lassen. Die folgenden „Owasp Top Ten“ stellen unter Web-Sicherheitsexperten einen anerkannten Konsens dar, was die derzeit kritischsten Lücken in Web-Anwendungen betrifft:

Cross Site Scripting (XSS)

Cross Site Scripting (XSS) ist die mit Abstand am weitesten verbreitete Schwachstelle - sie betrifft nahezu jede Web-Anwendung (siehe auch „Schwachstellen in Web-Seiten“, BSI-Lagebericht 2009, Seite 38). XSS-Lücken treten dann auf, wenn eine Anwendung von einem Benutzer übermittelte Daten an den Browser zurückgibt, ohne zu prüfen, ob die Zeichen codiert dargestellt werden müssen. Das ermöglicht einem Angreifer, beispielsweise Javascript-Code im Browser eines Opfers zur Ausführung zu bringen. Dieser bösartige Code kann im Browser auf alle Informationen der betroffenen Website zugreifen. Hierzu zählen Session-Informationen, die in Cookies gespeichert sind, aber auch in Eingabefelder eingegebene Informationen wie Passwörter. Durch Javascript lassen sich auch Bestandteile einer Web-Seite modifizieren oder identisch nachbilden. Im Falle einer Login-Maske werden die eingegebenen Login-Daten zunächst an einen Angreifer weitergeleitet, und erst dann erfolgt der eigentliche Login-Vorgang. Die XSS-Schwachstelle dient oft als Werkzeug für den als „Phishing“ bekannt gewordenen Angriff, bei dem Web-Nutzer verleitet werden sollen, sensible Daten preiszugeben.

Injection Flaws

Injection-Fehler sind in Web-Anwendungen sehr verbreitet. Sie entstehen, wenn die Web-Applikation an sie übermittelte Daten ungeprüft als Programmcode verwendet. Injection-Schwachstellen gibt es in mehreren Ausprägungen – etwa Webscript Injection, OS Command Injection oder SQL-Injection. Letztere tritt am häufigsten auf: Dabei übermittelt ein Angreifer innerhalb eines Parameters (etwa eines Formularfelds) gültigen SQL-Code an die Web-Anwendung, die ihn dann ausführt. Das ermöglicht ihm, auf Daten in der Datenbank zuzugreifen, sich diese anzeigen zu lassen, zu manipulieren oder sogar zu löschen. Viele Datenbanken erlauben zudem das Ausführen von Systemkommandos, was die Gefahr einer feindlichen Übernahme des Datenbanksystems erhöht. Oft wird außer Acht gelassen, dass

Foto: Fotolia/eyewave

Datenbank-Server im Inneren einer IT-Landschaft stehen. Eine Injection-Schwachstelle ermöglicht es einem Angreifer somit, ein internes System anzugreifen – an vielen Sicherheitsmaßnahmen vorbei.

Malicious File Execution

Bei Web-Anwendungen besteht die Möglichkeit, Schaddateien auf den Web-Server zu laden und zur Ausführung zu bringen. Häufig werden Dateien einfach entgegengenommen und auf dem Web-Server gespeichert, ohne vorab die Gültigkeit zu prüfen. Möglich werden entsprechende Angriffe oft durch Upload-Funktionen innerhalb der Web-Applikation. Bleibt eine hinreichende Prüfung der hochgeladenen Dateien aus, kann ein Angreifer bösartigen Code auf dem Server der Anwendung platzieren. Schadcode wird dabei in die Datei eingebettet und durch einen anschließenden Aufruf zur Ausführung gebracht. Die Auswirkungen einer solchen Attacke sind unterschiedlich und hängen stark von der Konfiguration des Web-Servers ab: Es besteht die Gefahr einer Server-Übernahme, auch ein Angriff auf die Benutzer der Web-Anwendung ist möglich.

Insecure Direct Object Reference

Bei der Entwicklung von Web-Anwendungen werden oft Objektreferenzen verwendet, um auf ein bestimmtes internes Implementierungsobjekt zu verweisen. Dabei kann es sich um Dateien, Verzeichnisse, Datenbankeinträge oder digitale Schlüssel handeln. Bei einer Insecure-Direct-Object-Reference-Lücke ist die Objektreferenz auf diese Objekte manipulierbar. Ein Angreifer kann durch geschicktes Manipulieren unautorisiert auf Dateien und Inhalte zugreifen. Vor allem wenn Applikationen mit sensiblen Daten arbeiten, sind diese Attacken gefährlich. Konkret verwenden Angreifer meist manipulierte IDs oder Pfadangaben, um etwa fremde Datensätze aus der Datenbank auszulesen oder unautorisiert auf Dateien des Web-Servers zuzugreifen.

Cross Site Request Forgery (CSRF)

Bei dieser Angriffsvariante wird der rechtmäßig angemeldete Benutzer zum „Bauernopfer“, indem – ohne sein Wissen – von seiner authentisierten Web-Anwendungs-Session eine Anfrage durchgeführt wird. Möglich ist das beispielsweise, wenn sich der bei einer Web-Anwendung angemeldete User beim Verlassen derselben nicht abmeldet und beim weiteren Surfen auf eine vom Angreifer präparierte Seite gelangt. Der dort platzierte bösartige Code löst über die autorisierte Sitzung des Nutzers einen Angriff aus, indem er beispielsweise im Namen des Opfers, jedoch ohne sein Wissen eine Funktion (etwa eine Überweisung) in Gang setzt. Solche Attacken zu erkennen oder nachzuweisen ist extrem schwierig. CSFR ist die Variante von Angriffen auf Web-Anwendungen, die derzeit am schnellsten zunimmt.

Information Leakage and Improper Error Handling

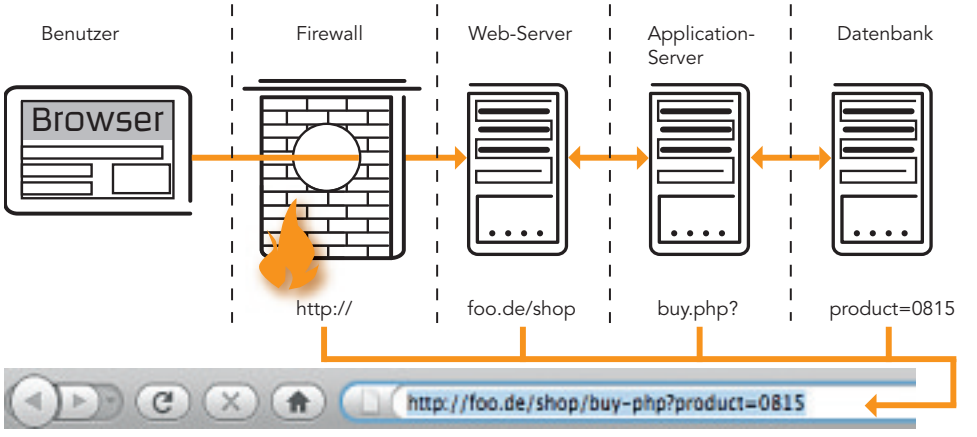
Web-Applikationen können unbeabsichtigt detaillierte Informationen etwa zum Aufbau der Anwendung oder zu verwendeten Softwareversionen preisgeben – beispielsweise über technische Fehlermeldungen, die direkt im Browser des Benutzers angezeigt werden. Solche Fehler kann ein Angreifer bewusst provozieren. Jedes System produziert eigene Fehlermeldungen, die Rückschlüsse auf Funktionsweisen und Eigenschaften der Web-Anwendung ermöglichen, aber ausschließlich den Entwicklern vorbehalten sein sollten. Besonders verbreitet ist die Identifikation aktuell verwendeter Softwareversionen: Wenn ein Angreifer über sie Bescheid weiß, kann er unter Umständen auch ihre Sicherheitslücken ausnutzen. Für den Betreiber einer Web-Anwendung ist diese Schwachstelle riskant, weil sich darauf basierend weiterführende Angriffe lancieren lassen. ▶

Was ist eigentlich...?

- **Webscript Injection:** Bei einer Webscript Injection versucht ein Angreifer, präparierten Scriptcode auf dem Web-Server zur Ausführung zu bringen.
- **OS Command Injection:** Bei dieser Angriffsvariante wird innerhalb von Formularfeldern ein Systemkommando untergebracht.
- **Man-in-the-Middle-Attacke:** Hier steht der Angreifer zwischen zwei Kommunikationspartnern und versucht, die ausgetauschten Daten aufzuzeichnen oder zu manipulieren.

Hinter der Firewall

Beim Aufruf einer Web-Seite im Browser werden die einzelnen URL-Segmente an die entsprechenden Server der Web-Anwendungen weitergereicht.



Quelle: OneConsult

Broken Authentication and Session-Management

Wenn in einer Web-Anwendung Zugangsdaten ausgetauscht werden, ist besondere Vorsicht geboten. Zugangs- und Sitzungsinformationen (Session-Tokens) sind oft nicht hinreichend vor Angreifern geschützt. Für Letztere sind Session-Tokens vor allem interessant, weil sie den Zugang zu gesperrten Bereichen unter dem Namen und mit den Rechten eines legitimen Benutzers ermöglichen. Schwachstellen im Standard-Authentisierungsmechanismus sind dabei nicht immer der einfachste Angriffspunkt. Vielmehr konzentrieren sich Hacker auf Sicherheitslücken in zusätzlichen Authentifizierungsfunktionen wie Logout-Funktion, Passwort-Erinnerung oder „Secret Questions“, um an Zugangsinformationen zu gelangen. Beim fehlerhaften Umgang mit Session-Tokens kann der Angreifer mittels XSS oder Man-in-the-Middle-Attacken in Besitz des Session-Tokens gelangen und so die Sitzung übernehmen (Session-Hijacking). Benutzer und Betreiber sind hiervon gleichermaßen betroffen, da sich neben Benutzer- auch Administratorenkonten kompromittieren lassen.

Insecure Cryptographic Storage

In vielen Web-Anwendungen fehlen kryptografische Funktionen oder sind schlecht implementiert. Dabei sind die Verschlüsselung schützenswerter Daten wie Zugangs- und Kreditkarteninformationen oder die Nicht-Vorhersagbarkeit etwa von Session-Tokens wichtige Mechanismen zur Absicherung einer Web-Applikation. Allerdings gewährleisten kryptografische Funktionen nicht zwingend, dass Daten und Tokens auch wirklich geschützt sind. Mitunter werden eigenentwickelte, schwache (etwa SHA-1 oder MD5) oder fehlerhaft implementierte Algorithmen verwendet, die gebrochen werden können und somit keinen Schutz bieten. Zudem werden die kryptografischen Schlüssel häufig an unsicheren Stellen (etwa in der Anwendung herunterladbar) aufbewahrt, wodurch die Sicherheit des gesamten Verschlüsselungsmechanismus aufgehoben wird.

Insecure Communications

Bei dieser Art von Schwachstelle werden sensible Daten über einen unsicheren Kommunikationskanal im Klartext oder nur teilweise beziehungsweise unsicher verschlüsselt übertragen. In diesen Fällen kann ein Angreifer durch passives Mitlesen oder eine Man-in-the-Middle-Attacke auf die transferierten Daten zugreifen. Vor allem bei der Übermittlung vertraulicher Informationen wie Zugangs-, Zahlungs- oder Kundendaten ist es notwendig, die Datenkommunikation zu verschlüsseln. Jedoch gibt es Web-Anwendungen, die nur den Austausch der Login-Informationen über einen sicheren Übertragungskanal abwickeln und die anschließende Kommunikation unverschlüsselt lassen. Dabei wird häufig vergessen, dass auch bei den folgenden Anfragen sicherheitsrelevante Authentifizierungsinformationen wie Session-Tokens übertragen werden. Das ermöglicht dem Angreifer, unverschlüsselte Tokens mitzulesen und sich mit der fremden Identität bei der Web-Anwendung zu authentisieren.

Failure to Restrict URL Access

Kritische Informationen in einer Web-Anwendung werden häufig lediglich dadurch geschützt, dass die entsprechende URL einem unautorisierten Benutzer nicht angezeigt wird oder nicht bekannt ist. Für eine Attacke lässt sich das ausnutzen, indem die URL direkt angesprochen wird. Die bekannteste Angriffsmethode, die diese Lücke missbraucht, nennt sich „Forced Browsing“: Der Angreifer versucht, durch systematisches „Ausprobieren“ ungeschützte Seiteninhalte oder Anwendungsfunktionen zu identifizieren und darauf zuzugreifen. Das Ziel ist häufig, versteckte Dateien oder URLs ausfindig zu machen, die bei der Implementierung der Berechtigungen übersehen wurden. Für den Betreiber einer Web-Anwendung sind solche Schwachstellen besonders brisant, da ein Angreifer Informationen über deren Aufbau und Struktur gewinnt oder sogar Zugriff auf administrative Funktionen der Seite erhält.

Fazit

Die Owasp Top Ten liefern einen umfassenden Überblick über die aktuell gefährlichsten Schwächen in Web-Anwendungen. Zu den einzelnen Sicherheitslücken gibt es auf der Web-Seite der Organisation (www.owasp.org) neben detaillierten Beschreibungen Bewertungen der jeweiligen Gefahren sowie einen entsprechenden Maßnahmenkatalog. Owasp, das sein Top-Ten-Projekt primär als „Aufklärungs-Dokument“ definiert, rät, Sicherheitsfragen beim Entwickeln einer Web-Anwendung von Beginn an zu beachten. Dazu stellt die Gruppe zahlreiche weitere unterstützende Projekte bereit (etwa Development Guide, Code Review Guide, Testing Guide). Sämtliche Informationsmaterialien und Tools, die von Owasp zur Verfügung gestellt werden, sind unter anerkannten Free-and-Open-Source-Software-Lizenzen veröffentlicht und somit für jeden frei zugänglich. *(kf)*

***Christoph Wolfert** ist Security-Consultant bei OneConsult Deutschland.