

SOA-Security mit ONR 17700

Die ONRegel 17700 definiert klare Vorgaben für die sichere Entwicklung und Beschaffenheit von Webapplikationen. Das Regelwerk lässt sich auch für Webservices einer Service-oriented Architecture (SOA) anwenden und ermöglicht dann auch deren Zertifizierung.

Von Thomas Kerbl, Wien

Bei klassischen, in sich abgeschlossenen Applikationen ist die Anzahl der nach außen sichtbaren Schnittstellen recht überschaubar. Durch die Offenheit einer serviceorientierten Architektur (SOA) werden hingegen viele Schnittstellen sichtbar, die zuvor innerhalb der Applikation verborgen waren. Eine SOA-Implementierung lässt sich mit einem objekt-orientierten Programm vergleichen: Dabei stellt jeder Service ein Objekt dar, das unterschiedliche Methoden beziehungsweise Funktionen anbietet. Bei einer abgeschlossenen Applikation können diese Funktionen nur von der Anwendung selbst genutzt werden – in einer SOA ist es hingegen erwünscht, dass viele Systeme möglichst viele Funktionen nutzen können (vgl. Abb. 1).

Während also auch ein Angreifer auf die Methoden und Funktionen der einzelnen Objekte einer klassischen Applikation nur über die definierten Frontends zugreifen kann, hat er in einer SOA-Umgebung Zugriff auf alle Services, die er im Netzwerk sieht. Während der Zugriffsschutz in der geschlossenen Applikation vom Design her vorgegeben ist, muss man ihn in einer SOA explizit implementieren. Dies wird in der Realität dadurch erschwert, dass eine Vielzahl der Services von mehreren anderen Services und Applikationen genutzt wird.

Das SOA-Konzept bringt somit völlig neue Anforderungen an das Security-Design mit sich. Es müssen jedoch nicht nur neue Bedrohungen gekontert werden, es bieten sich auch neue Möglichkeiten, um zentralisierte Schutzmaßnahmen in Form von Services einzusetzen (s.u.). Hilfestellung und Orientierung für richtiges Design und für ein Sicherheits-Audit kann man der Regel 17700 „Informationsverarbeitung - Sicherheitstechnische Anforderungen an Webapplikationen“ des Österreichischen Normungsinstituts (ON) entnehmen[1].

Angriffsvektoren

Webservices verhalten sich sehr ähnlich wie Webapplikationen. Wesentliche Unterschiede finden sich vor allem in den eingebetteten Kommunikationsprotokollen und in der Aufbereitung der Ausgabe. Ein Gutteil der Bedrohungen, denen

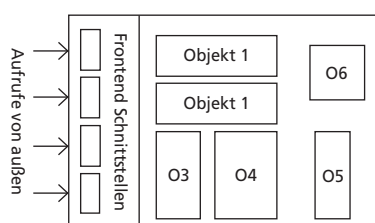
Webapplikationen ausgesetzt sind, ist daher direkt auf Webservices übertragbar.

Vor allem Injection-Angriffe wie SQL Injection, LDAP Injection oder XPATH Injection sind auch für Webservices eine reale Bedrohung. Bei derartigen Attacken werden Parameter manipuliert, die von der Applikation in einer Abfrage an ein Backend-System verwendet werden. Auf diesem Weg ist es unter Umständen für einen Angreifer möglich, beispielsweise Passwörter aus einem LDAP-Verzeichnis auszulesen.

Pufferüberläufe können auch in Webservices ein Problem darstellen. Selbst wenn sie mit typischeren Sprachen wie Java implementiert sind, können etwa verwendete C-Module über das Java Native Interface (JNI) für Buffer Overflows anfällig sein und somit die Gesamtsicherheit des Webservices gefährden. Schnittstellen zu Legacy-Systemen können ebenfalls ein Risiko darstellen, wenn keine entsprechenden Filtermechanismen zum Einsatz kommen.

Webservices, die über XML kommunizieren, sind zudem von so genanntem XML Entity Bombing bedroht: Dieser Angriff basiert auf dem ENTITY Tag von XML, das die Definition von Variablen ermöglicht – und diese auch zusammensetzen kann. Versucht ein Angreifer mehre-

geschlossene Applikation



Applikation in einer SOA

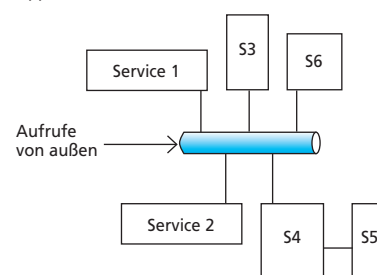


Abbildung 1: Vergleich der Schnittstellen von klassischen Applikationen und SOA

re Variablen in einer endlosen Schleife rekursiv zusammenzufügen, wird der Webservice ohne entsprechende Gegenmaßnahmen abstürzen; dies ermöglicht zumindest einen Denial-of-Service-Angriff. Es gibt unterschiedliche Varianten von Entity Bombs, die alle verhindert werden müssen.

Replay-Attacken stellen ein weiteres Bedrohungsfeld dar. Selbst wenn die gesamte Kommunikation signiert erfolgt, kann ein Angreifer ein Paket eventuell abfangen und zu einem späteren Zeitpunkt selbst erneut versenden: Die Signatur ist gültig und der Vorgang wird erneut angestoßen, obwohl der Angreifer keine Berechtigung für das abgesetzte Kommando hat. Ohne entsprechende Gegenmaßnahmen können Replay-Attacken die Applikationslogik leicht umgehen. So könnte zum Beispiel eine Überweisung abgefangen und einige weitere Male an den verarbeitenden Service gesendet werden.

Vorgaben der ONR 17700

An dieser Stelle kann zwar nur ein Auszug aus der langen Bedrohungsliste erscheinen, gegen die ein Webservice gerüstet sein muss – die ONR 17700 deckt jedoch ein sehr breites Bedrohungsspektrum ab.

Dazu definiert das Regelwerk eine Vielzahl an Anforderungen, die ein Webservice erfüllen muss, um als sicher bewertet werden zu können. Der Fokus liegt dabei auf den Berei-

chen Architektur, Authentifizierung und Sitzungsmanagement, Benutzereingaben, Einbinden von Dateien, Ausführen externer Programme, Datei-Uploads und -Generierung, Datenbanken, System- und Fehlermeldungen sowie Kryptographie.

Security-Audits haben gezeigt, dass Webapplikation und Webservices, die alle Vorgaben der ONR 17700 erfüllen, selbst von erfahrenen Angreifern nur mit unverhältnismäßig großem Aufwand kompromittiert werden können. Als Beispiel für Schutzmaßnahmen kann hier der Einsatz von Kryptographie gegen Replay-Attacken dienen: Ein kryptographisch gesicherter Kommunikationstunnel kann sicherstellen, dass keine Pakete abgefangen und nachträglich dupliziert werden. Zudem können Zeitstempel dabei helfen, duplizierte Pakete als solche zu erkennen.

Um ein zweites Beispiel zu geben: ONR 17700 definiert, dass jegliche Benutzereingaben gefiltert werden müssen – hierin ist eine Gegenmaßnahme zu Injection-Attacken zu sehen, die auf ungefilterten Input zurückzuführen sind. Bei einem Webservice ist hierzu eine kontextsensitive Filterung einzusetzen, was eine Anpassung der Filter abhängig von der Verwendung des Benutzerinputs bedeutet – eine Datenbankabfrage erfordert naturgemäß andere Filter als eine LDAP-Anfrage. Werden etwa Daten eines Benutzers aus dem LDAP-Verzeichnis ausgelesen und der Benutzername zuvor vom Client erfragt, so müssen vor der Übergabe an das Backend-System spezifische Filter sicherstellen, dass keine unerwünschten Anfragen möglich werden.

Durch die Verkettbarkeit von Webservices können viele Anforderungen auch in eigene Services ausgelagert werden: Um beispielsweise einer SQL Injection entgegenzuwirken, muss man den entsprechenden Filtermechanismus nicht in jeden

einzelnen Webservice integrieren, der Datenbank-Funktionen enthält. Alternativ dazu ließe sich ein zentraler Service für Datenbankzugriffe implementieren, der die Zugriffe abstrahiert und die entsprechenden Filterungen vornimmt. Sollten zu einem späteren Zeitpunkt Änderungen am Filter vorgenommen werden müssen, genügt dann die Änderung an einer einzigen Stelle. Würde der Filter hingegen in jedem Service enthalten sein, müsste man auch alle Services anpassen. Dieses Konzept sollte bereits beim Design einer SOA Umgebung Berücksichtigung finden.

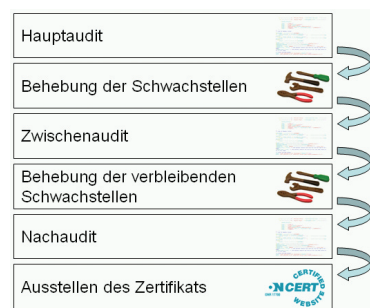
Zertifizierung

Um Webservices gemäß ONR 17700 zu zertifizieren, wird ein mehrstufiges Sourcecode-Review angewendet. Nachdem die Entscheidung für eine solche Zertifizierung gefallen ist, führt der erste Weg zum Österreichischen Normungsinstitut (www.on-norm.at), das einen akkreditierten Auditor vermittelt, der die Evaluierung durchführt.

Zuerst wird hierzu der gesamte Sourcecode auf Schwachstellen untersucht. Erkannte beziehungsweise vermutete Mängel werden in einer Teststellung verifiziert. Handelt es sich um ausnutzbare Schwachstellen, werden diese vermerkt und gesammelt am Ende des Audits an das Entwicklerteam kommuniziert. Dieser erste Schritt wird als Hauptaudit bezeichnet und stellt den größten Block des Zertifizierungsprozesses dar (vgl. Abb. 2).

Das Entwicklerteam hat dann die Möglichkeit, die gefundenen Schwachstellen zu beheben. Abhängig vom Aufbau des Services kann dies durch die Änderung weniger Zeilen Code geschehen oder einen grundlegenden Umbau erfordern. Es wird daher empfohlen, schon in der Designphase ein Security Framework vorzusehen, um sicherheitsrelevante Änderungen

Abbildung 2:
Ablauf einer
Zertifizierung
nach ONR 17700



zentral an einer Stelle durchführen zu können (z.B. Erweiterung eines Inputfilters).

Um die korrekte Umsetzung der Änderungen zu verifizieren, werden alle geänderten Dateien einem zweiten Audit unterzogen. Die Änderungen werden auch auf die Teststellung übertragen, um dem Auditor die Möglichkeit zu geben, die theoretischen Angriffsvektoren in der Praxis zu überprüfen. Sollten Mängel in der Umsetzung vorliegen, unterrichtet der Auditor erneut die Entwickler, damit diese letzte Probleme im Sourcecode beheben können.

Bisherige Zertifizierungsprojekte haben gezeigt, dass im anschließenden Nachaudit nur noch selten verbliebene Schwachstellen zu finden waren – die Applikation konnte dann meist als sicher bewertet werden. Sobald dieser Status erreicht und vom Auditor beglaubigt ist, stellt das Österreichische Normungsinstitut (ON) das Zertifikat aus. Um die zertifizierte Version eindeutig bestimmen zu können, werden beim ON Checksummen aller zertifizierten Dateien hinterlegt.

Fazit

Eine Zertifizierung nach ONR 17700 bringt Entwicklern und Zulieferern von Webservices einen deutlichen Marktvorteil: Sie sind in der Lage die Qualität ihres Produkts hinsichtlich von Sicherheitsaspekten zu belegen. Zahlreiche Unternehmen fordern bereits Compliance mit ONR 17700 in ihren Beschaffungsrichtlinien. Selbst wenn keine Zertifizierung des Produktes erfolgt, sollte der Hersteller eines Webservices auf alle Fragen des Regelwerks eine gute Antwort geben können.

Breibt man selbst eine SOA im Unternehmen, liefert eine Zertifizierung der Services die Gewissheit, dass selbst bei Schwachstellen in den Frontend-Applikationen die

Backend-Services nicht so leicht kompromittiert werden können. So könnte beispielsweise ein zertifizierter Webservice, der Zugriffe auf eine Datenbank abstrahiert, sicherstellen, dass keine Daten über SQL Injection auszulesen oder zu verändern sind, selbst wenn ein Angreifer den Input-Filter des Frontends überlistet.

Ein weiterer Anwendungsfall der ONR 17700 und insbesondere ihres Kapitels 6 „Formulare und Benutzereingaben“ ist übrigens die Abwehr von Cross-Site-Scripting- und Phishing-Angriffen auf Web-Dienste.

Angesichts des großen Bedrohungspotenzials, sollte man zumindest bei businesskritischen Webservices und -applikationen eine Zertifizierung verlangen, um Gewissheit über das Sicherheitsniveau der Implementierung zu haben. ■

Thomas Kerbl ist Senior Security Consultant und ONR-17700-Auditor bei der SEC Consult Unternehmensberatung GmbH (www.sec-consult.com).

Literatur

[1] ON Österreichisches Normungsinstitut, ONR 17700 „Informationsverarbeitung - Sicherheitstechnische Anforderungen an Webapplikationen“, Juni 2005, 16,30 €, erhältlich über ON-Webshop www.on-norm.at/ecom/

[2] ON Österreichisches Normungsinstitut, Thema „Information Security“, www.on-norm.at/publish/1801.html

