APEX geht schnell, aber ist es auch sicher?

Anja Hildebrandt
buw Holding GmbH & Co. KG
Osnabrück

Schlüsselworte

APEX, Security, Authentifizierung, Autorisierung, SQL Injection, Cross Site, Scripting, Session State Protection

Einleitung

Gerade als APEX-Einsteiger ist man begeistert von den Möglichkeiten, die Oracle APEX bietet, um moderne benutzerfreundliche Webanwendungen in Rekordzeit zu erstellen. Sicherheitsaspekte kommen hierbei jedoch meist zu kurz.

Diese Präsentation richtet sich an Einsteiger und zeigt, worauf es bei der Erstellung einer sicheren Anwendung ankommt und welche Bausteine APEX bereits mitbringt, um häufig auftretende Sicherheitslücken von Beginn an zu schließen.

Anhand von Beispielen wird insbesondere auf die Wirkungsweise der standardmäßig verfügbaren Authentifizierungs- und Autorisierungsmechanismen in APEX eingegangen.

Ein weiteres Augenmerk wird auf SQL Injection und Cross Site Scripting liegen. Es wird demonstriert, welche üblichen Fehler in der Entwicklung Angriffe dieser Art begünstigen und welche Techniken genutzt werden können, um die eigene Anwendung entsprechend zu schützen.

Hierbei gilt es zu beachten, das ein vollständiger Überblick aller Sicherheitsaspekte von APEX den Rahmen dieser Präsentation sprengen würde. Aus diesem Grund wird der Fokus auf einigen der größeren Punkte liegen, die jeder Entwickler kennen und in seiner täglichen Arbeit nutzen sollte.

Ist APEX sicher?

APEX selbst wird von Oracle einem rigorosen Testprozess ausgesetzt, der für höchstmögliche Stabilität und Sicherheit sorgt. Es ist inzwischen kein neues oder junges Produkt mehr. APEX ist erwachsen geworden. Sicherheitslücken wurden geschlossen und so wurden auch die Anwendungen, die wir erstellen, automatisch weiter gehärtet.

Allerdings ist es für Entwickler immer noch möglich, unsichere Anwendungen zu bauen. Das passiert normalerweise dann, wenn sich der Entwickler der vorhandenen Bedrohungen nicht bewusst ist und nicht weiß, welche Features er nutzen kann, um diesen Bedrohungen entgegen zu wirken.

APEX bietet eine ganze Reihe an Sicherheitsfeatures out of the box. Einige dieser Features findet man auch in anderen Produkten, andere nicht. Die meisten dieser Optionen wie Authentifizierungs- und Autorisierungsschemata, Session State Protection und viele andere Sicherheitsattribute sind einfach und deklarativ nutzbar.

Zugangskontrolle

Beschäftigen wir uns zunächst mit Zugangskontrollmechanismen. Hierbei unterscheidet man zwischen Zugriff auf Anwendungsebene und Datenebene.

Auf Anwendungsebene betrachten wir, wer auf die Anwendung zugreifen kann. Wer kann welche Funktionalitäten innerhalb der Anwendung nutzen? Wer hat Zugriff auf eine bestimmte Komponente? Das kann eine Region, eine Seite, ein Item oder ein Button sein. Hierfür werden in APEX Authentifizierungs- bzw. Autorisierungsschemata benutzt.

Die andere Seite stellen die Daten dar. Welche Datensätze darf ein User tatsächlich sehen?. Hierfür werden meistens einfache WHERE-Klauseln genutzt.

Authentifizierungsschemata

Der erste Ansatzpunkt zur Sicherung einer Anwendung sind die Authentifizierungsschemata. Sie dienen der Identifizierung des Users.

Typischerweise erfolgt die Identifizierung eines Users in einer Webanwendung über eine Kombination von Username und Passwort. Häufig wird auch auf einen Single Sign On Service gesetzt. Im Prinzip kann die Identifizierung des Users aber durch eine Vielzahl von Optionen (von Sicherheitskarte bis Retinascan) durchgeführt werden.

APEX wird mit einer Reihe vorkonfigurierter Schemata (siehe Abb. 1) ausgeliefert, die sofort einsatzbereit sind. Ein häufig verwendetes Schema ist LDAP. Wenn also bereits ein Active Directory Server im Einsatz ist, kann die neue Anwendung einfach angebunden werden.

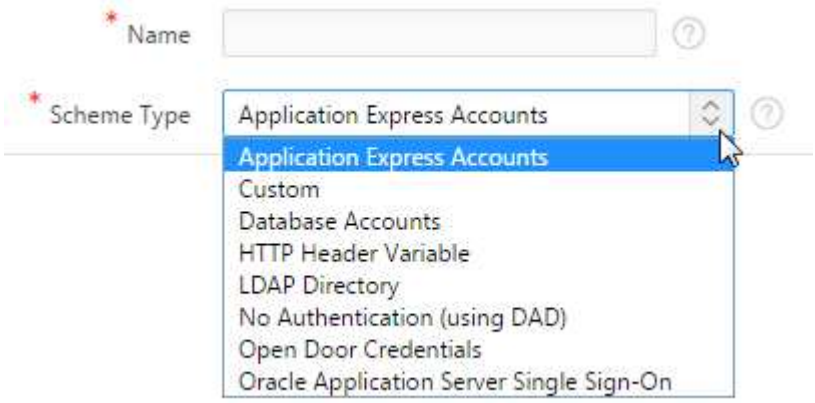

Abb. 1 : Typen von Authentifizierungsschemata

Eine weitere Option ist eine HTTP Header Funktion, die eine simple Integration ermöglicht, sofern bereits eine Single Sign On Lösung im Einsatz ist, die HTTP Header Variablen setzt.

Ein weiteres interessantes Schema ist das Open Door Schema, das leider vielen Entwicklern nicht bekannt ist. Es ermöglicht es eine Anwendung als beliebiger Endnutzer zu testen. Die Anmeldung erfordert nur einen Usernamen, aber kein Passwort. Natürlich sollte dieses Schema nur in der Entwicklungsumgebung zum Einsatz kommen.

Autorisierungsschemata

Der nächste wichtige Bereich, den es sich anzuschauen lohnt, sind Autorisierungsschemata. Sie erlauben es uns genau festzulegen, welche Komponenten und Funktionen ein User innerhalb der Anwendung nutzen darf. Typischerweise wird dies über eine Gruppenzugehörigkeit gelöst.

In der Demo werden wir der Einfachheit halber die User- und Gruppenfunktionalität der APEX Umgebung nutzen.

Natürlich kann eine Autorisierung erst erfolgen, wenn der User zuvor authentifiziert wurde. Man muss wissen, wer ein User ist, bevor man herausfinden kann, was er innerhalb des Systems tun darf.

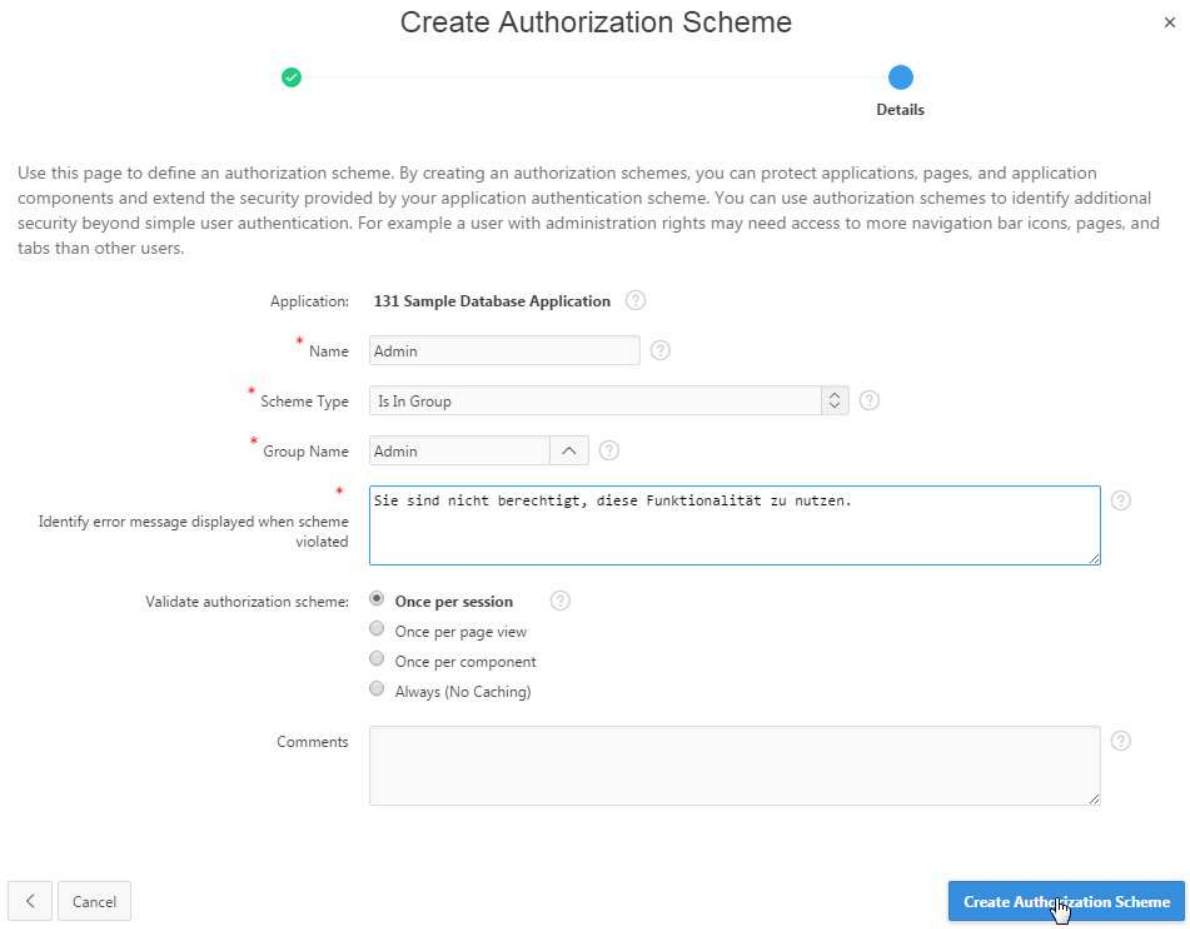

Abb. 2 : Autorisierungsschema anlegen

Die Autorisierungsschemata in APEX sind sehr flexibel. Sie können beliebig konfiguriert werden. Typischerweise basieren sie auf einem Rollensystem, d.h. es wird ein eigenes Autorisierungsschema für jede Rolle angelegt, die im System vorgesehen ist.

Über diesen Mechanismus kann der Entwickler die Zugriffe auf alle notwendigen Komponenten – von Seiten bis hinunter zu einzelnen Elementen - steuern.

Conditions vs Authorization

Einige denken sich nun vielleicht: „Moment mal. Ich nutze vielleicht keine Autorisierungsschemata, aber Conditions. Was ist denn da der Unterschied?“

Der Unterschied ist klein aber entscheidend.

Im Allgemeinen wird empfohlen, Conditions für allgemeine Zwecke zu nutzen. Beispielsweise ist es üblich, das die Button „Delete“ und „Apply Changes“ in einem Formular ausgeblendet werden, wenn sich das Formular im Create-Modus befindet. Das wird durch Conditions erreicht. Sie sind meist mehr Business Logik als alles andere.

Autorisierungschemata sollten hingegen genutzt werden, wenn es um sicherheitsrelevante Bereiche geht. Wenn also beispielsweise ein Button ausgeblendet werden soll, weil der User kein Admin ist, dann wäre das ein Fall für ein Autorisierungsschema.

Aber was genau muss man denn nun sichern?

Der Weg zum Ziel – Erst das Ziel, dann der Weg…

Nehmen wir einmal an, dass wir eine Seite der Anwendung nur für eine bestimmte Nutzergruppe freigeben wollen. Die meisten Entwickler würden nun alle Links und Buttons die zu dieser Seite verlinken mit einer Autorisierung versehen. Aber was passiert, wenn an der Adresszeile des Browsers oder am Code des Links herum manipuliert wird? Unter Umständen erhält der User dann doch Zugriff.

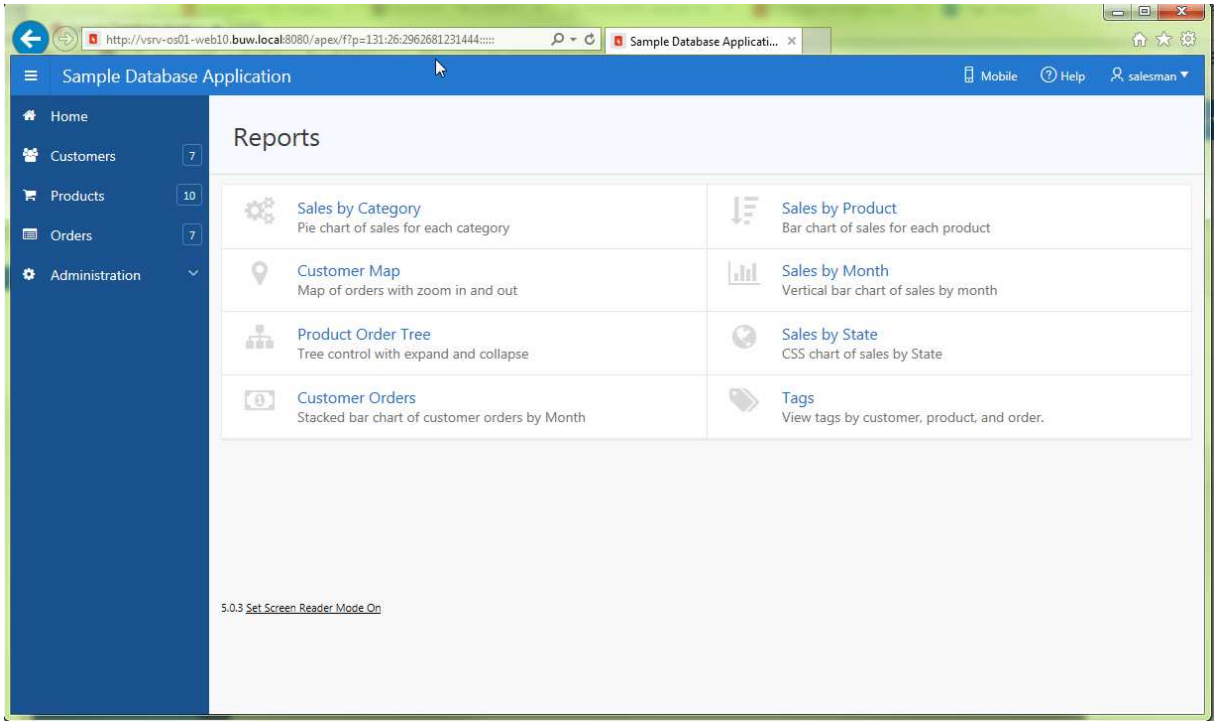

Abb. 3 : Aufruf der Reportsseite per URL-Manipulation

Um dies zu verhindern ist es nötig nicht nur den Weg zum Ziel, sondern vor Allem auch das Ziel selbst zu schützen.

Wenn es um Autorisierung geht, ist es daher ratsam, immer erst das Ziel zu sichern und dann den Weg oder die Wege dorthin.

Was bedeutet das?

Schaut man sich beispielsweise Reiter, Button u.Ä. an (eigentlich alles, was einen Link oder Aufruf enthält), erkennt man das sie nur der Weg zum Ziel sind – der Weg der Navigation in diesem Fall.

Was tatsächlich gesichert werden muss, ist das Ziel, also die Seite zu der der Link führt oder der Prozess, der durch den Button ausgelöst wird.

Konstruieren wir ein weiteres Beispiel: In einem Report werden jedem User nur genau die Datensätze angezeigt, die er selbst erstellt hat. Dies wird durch das Ergänzen einer entsprechenden WHERE Bedingung im Report erreicht. Der User kann also eigentlich keine Datensätze anderer User öffnen und bearbeiten. Aber auch hier könnte man durch Manipulation der URL unberechtigten Zugriff erlangen.

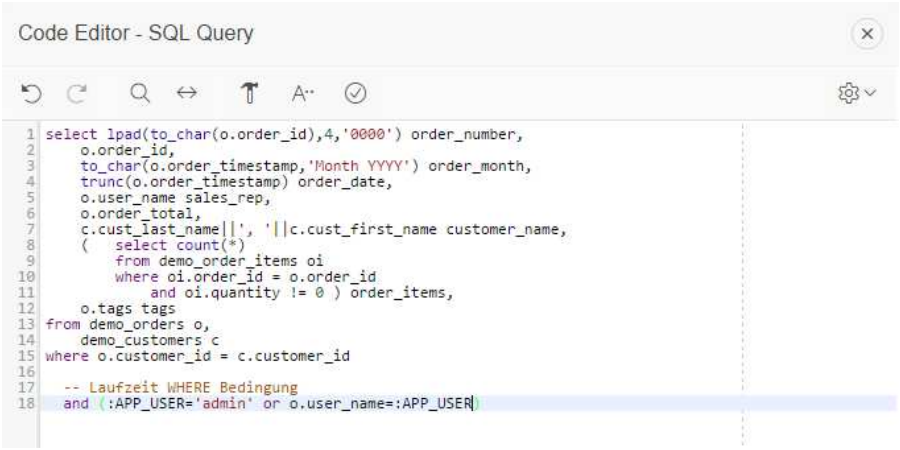

Abb. 4 : Report mit User-bezogener WHERE Bedingung

Um diesen Manipulationen Einhalt zu gebieten, gibt es zwei Möglichkeiten.

Laufzeit WHERE Bedingung

Die erste Möglichkeit sind Laufzeit WHERE Bedingungen. Wir haben bereits gesehen, das das ergänzen einer WHERE Bedingung im Report nicht ausgereicht hat unsere Daten vor unberechtigtem Zugriff zu schützen. Der Report ist auch hier nur der Weg und nicht das Ziel.

Was wir tatsächlich sichern müssen, ist die Seite, die der Bearbeitung der Orders dient. Wir müssten also dort dieselbe WHERE Bedingung hinterlegen wie im Report. Eine Möglichkeit dies in APEX zu tun sind Laufzeit WHERE Bedingungen.

Abbildung 5 zeigt eine entsprechende Einstellung.

Ein weiterer Weg wäre die Verwendung von Virtual Private Database. Diese Option erfordert etwas mehr Setupaufwand ist aber in jede Fall die nützlichste Lösung. Da für VPD aber eine Enterprise Edition erforderlich ist, werden wir uns in der Demo auf die erste Möglichkeit konzentrieren.

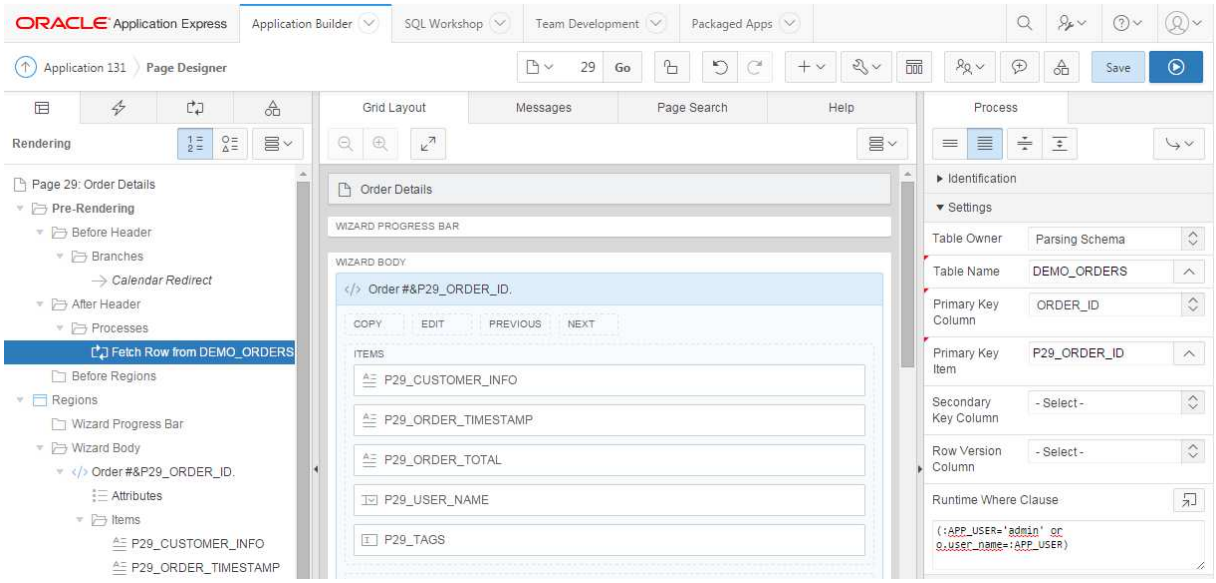

Abb. 5 : Runtime WHERE Bedingung

Session State Protection

Es gibt allerdings einen weit eleganteren Weg in APEX, die URL zu schützen. Session State Protection wurde genau zu diesem Zweck implementiert. Sie verhindert, das die Item und Value Bereiche der URL manipuliert werden können (Position 7 und 8 der APEX URL).

Beim Anlegen einer neuen Anwendung ist das Feature in den aktuellen Versionen standardmäßig aktiviert. In der Demo werden wir uns die verfügbaren Level der Session State Protection genauer ansehen.

Weitere Bedrohungen

Weitere Bedrohungen, die Web-Anwendungen regelmäßig angreifbar machen, sind SQL Injection und Cross Site Scripting.

In einer Demo werden wir uns ansehen, wie leicht es ist, eine Webseite auf ihre Angreifbarkeit durch SQL-Injection zu testen. Hierfür kommt das SQLMap Framework zum Einsatz.

Es wird außerdem demonstriert, wie derartige Probleme zum Beispiel durch den Einsatz von DBMS_ASSERT verhindert werden können.

Kontaktadresse:

Anja Hildebrandt

buw Management Holding GmbH & Co. KG

Rheiner Landstraße 195

D-49078 Osnabrück

Telefon:	+49 (0) 541-9462 22827
Fax:	+49 (0) 541-9462 987
E-Mail	anja.hildebrandt@buw.de
Internet:	www.buw.de