Zusammenfassung: Alle Oracle Forms Anwendungen sind per Default durch SQL Injection angreifbar.
Oracle Applications >=11.5.9 ist davon nicht betroffen, da hier standardmäßig der Wert FORMSxx_RESTRICT_ENTER_QUERY auf TRUE gesetzt ist.
(VU#718548)

Über Oracle Forms: Oracle Forms 10*g* ist Oracles preisgekröntes Web Rapid Application Development Tool, Teil der Oracle Developer Suite 10*g*. Es ist ein hoch produktives, End-to-End, PL/SQL basierende, Entwicklungsumgebung zum Entwickeln von firmenweiten, datenbankzentrierten Internetanwendungen.
Oracle Application Server 10*g* liefert Out-of-the-box eine optimierte Webentwicklungsplattform für Oracle Forms 10*g*. Oracle selbst benutzt Oracle Forms zur Entwicklung von Oracle Applications.

Betroffene Produkte: Alle Versionen von Oracle Forms (3.0-10g, C/S und Web), Oracle Clinical, Oracle Developer Suite

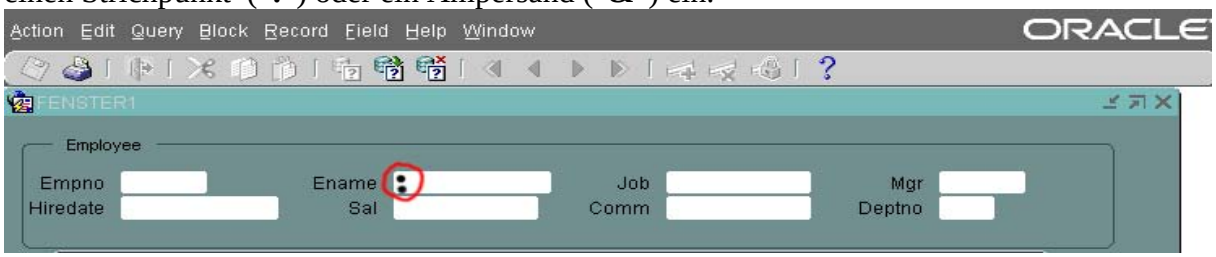
Korrektur: Setzen Sie die undokumentierte Umgebungsvariable **FORMS**xx**_RESTRICT_ENTER_QUERY=true (FORMS60_RESTRICT_ENTER_QUERY for Forms 6.x, FORMS90_RESTRICT_ENTER_QUERY for Forms 9.x/10g)** und starten Sie den Forms Server neu. Diese Umgebungsvariable deaktiviert die Möglichkeit die Query/Where-Funktionalität zu nutzen.

Wenn Sie die Query/Where-Funktionalität in Ihrer Anwendung wirklich benötigen, können Sie alternativ dazu

einen PRE_QUERY/ON-ERROR-Trigger für JEDES Eingabefeld schreiben und die gesamte Benutzereingabe für jedes Oracle Forms Modul (*.fmb) validieren.

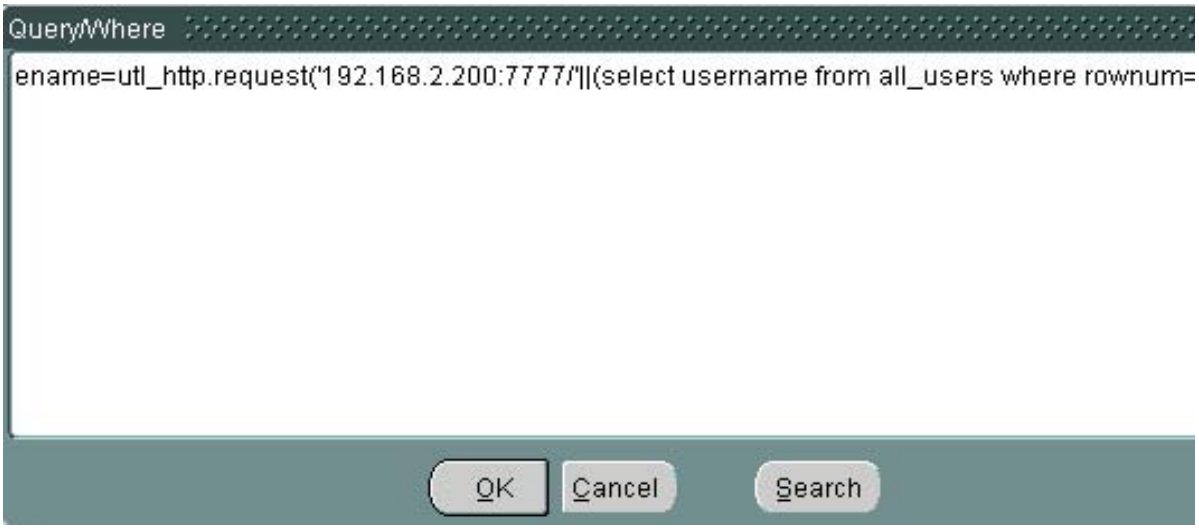
Hintergrund:
Es gibt in Oracle Forms (C/S und Web) ein (altes, oft vergessenes) Feature genannt "Query/Where", dass es jedem Benutzer gestattet, ein existierendes SQL Statement zu verändern. Für Poweruser ist dies ein sinnvolles und mächtiges Feature. Dementsprechend ist es aber auch gefährlich, da jeder Oracle Forms Benutzer jedes SQL Statement ausführen darf.

Kurze Demonstration von SQL Injection in Oracle Forms
1. Starten Sie ein Forms Modul, wechseln Sie in den Abfrage-Modus und geben Sie dann einen Strichpunkt ("**:**") oder ein Ampersand ("**&**") ein.



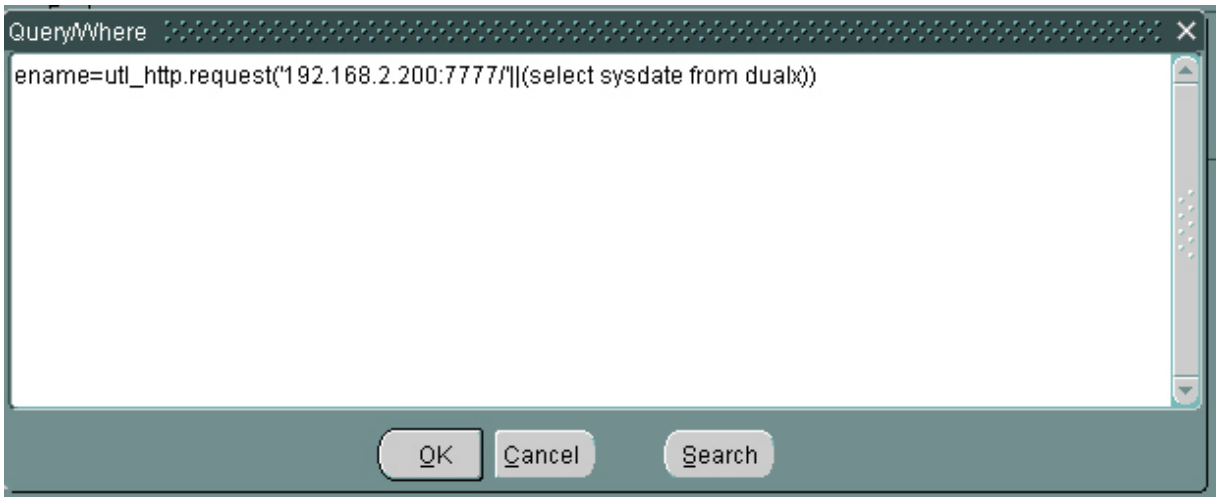
2. Ein leeres Query/Where Fenster erscheint

3. Eingabe eines SQL Statements
Das folgende Statement sendet das Ergebnis der SQL Abfrage "**select username from all_users where rownum=1**" zu einem externen (oder internen) Webserver. Bitte beachten Sie, dass utl_http.request nur eine Rückgabezeile gestattet.



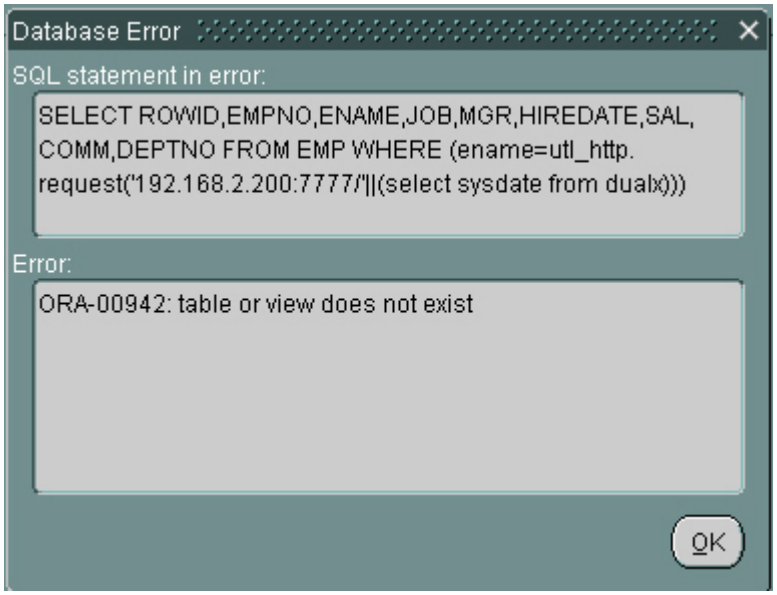
```
192.168.2.200 - - [14/Feb/2005:10:42:20 +0100] "GET /SYS HTTP/1.1" 404 209
```

Falls Ihr SQL Statement inkorrekt ist, wie in dem folgenden Beispiel,



liefert Oracle eine Fehlermeldung "**FRM-40505: Oracle error: unable to perform query.**" zurück

Diese detaillierte Fehlermeldung kann man sich durch Auswahl von "Display Error" im Hilfe-Menü ansehen.



Auswirkungen:
Die genauen Auswirkungen dieser SQL Injection Lücke hängt von der Architektur der Oracle Forms Anwendung ab.

Besitzt derjenige Oracle Benutzer, der in der Forms Anwendung verwendet wird, DBA Privilegien (wie z.B. in Oracle Applications), kann JEDER Forms Benutzer ALLE Daten in der Datenbank sehen.
Setzen Sie in einer existierenden Oracle Applications Umgebung den Wert FORMSxx_RESTRICT_ENTER_QUERY niemals auf FALSE, da sonst JEDER Benutzer ALLE Daten sehen kann.

Wenn Ihre Forms Anwendung ein eigenes Benutzer oder Mandantenkonzept implementiert ist es möglich, dass Benutzer Daten sehen können (z.B. die eines anderen Mandanten), die nicht für Sie bestimmt sind.

In allen anderen Fällen kann ein Forms Benutzer immer noch alle PL/SQL Packages ausführen, die an Public zugewiesen sind.

Korrektur:
Es gibt zwei unterschiedliche Möglichkeiten, dieses Problem zu lösen:

(a) Deaktivieren Sie die Query/Where Funktionalität durch Setzen der undokumentierten Umgebungsvariablen (möglich seit mit Forms 6.0.8.18.0)
FORMSxx**_RESTRICT_ENTER_QUERY=true**
(xx=60 für Forms 6.x, xx=90 für Forms 9.x/10g)
und starten Sie den Forms Server neu.

Überprüfen Sie, ob die Query/Where-Funktionalität nun deaktiviert ist.

(b) Schreiben Sie einen PRE_QUERY und einen ON-ERROR Trigger für JEDES Eingabefeld um die Benutzereingaben zu validieren. Ein Beispiel dafür finden Sie in der Metalink Note 163305.1.

Bitte beachten Sie aber, dass die in Metalink vorgeschlagene Lösung unvollständig (und damit unsicher) ist, da die Überprüfungen für **$** und **#** fehlen.

Referenzen:

- Metalink Document 163305.1: How to Disable the Query/Where in Forms
- Critical Patch Update - April 2005
- Härten des Oracle Application Server 9i Rel1, 9i Rel.2 and 10g: http://www.red-database-security.com/wp/DOAG_2004_dt.pdf
- Google Hacking of Oracle Technologies (z.B. Oracle Forms): http://www.red-database-security.com/wp/google_oracle_hacking_us.pdf

Historie:

- 7-oct-2003 Oracle secalert informiert
- 7-oct-2003 Fehler bestätigt
- 12-apr-2005 Advisory Version 1.00 veröffentlicht
- 15-apr-2005 Advisory Version 1.01 CERT Vulnerability Nummer hinzugefügt

Andere Dokumente bzgl. Oracle Sicherheit:

Härten des Oracle Application Server 9i Rel.1, 9i Rel.2 und 10g:
http://www.red-database-security.com/wp/DOAG_2004_dt.pdf

Härten von Oracle DBA und Entwickler Arbeitsplätzen:
http://www.red-database-security.com/wp/hardening_admin_pc_dt.pdf

Datenbank Rootkits / Oracle Rootkits:
http://www.red-database-security.com/wp/db_rootkits_dt.pdf

Google Hacking of Oracle Technologies:
http://www.red-database-security.com/wp/google_oracle_hacking_us.pdf

Über Red-Database-Security GmbH:
Red-Database-Security GmbH ist auf Oracle Security spezialisiert. Wir bieten Oracle Sicherheitstrainings, Oracle Datenbank und Oracle Application Server Audits, Penetration Tests, Oracle (Security) Architektur Reviews und Softwarelösungen gegen Oracle Rootkits an.

Kontakt:
Bei Fragen oder Problemen können Sie uns unter

info at red-database-security.com

erreichen.