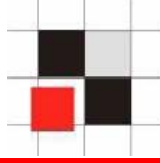
Härten des Oracle Application Server 9*i* Rel1, 9i Rel.2 und 10*g*

Alexander Kornbrust
10-Nov-2004

Überblick:

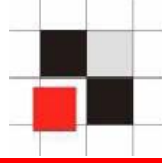


1. Einführung
2. Härten des Betriebssystems
3. Verwendete Architektur
4. Patching
5. Infrastrukturdatenbank
6. Apache
7. Webcache
8. Forms
9. Reports

Einführung

- Warum den Application Server schützen?
- Weil...
  - Sicherheit auf allen Ebenen einer Anwendung notwendig ist (OS, DB, iAS, Anwendung und Client)
  - Ein gehärteter Application Server oftmals weniger Patches benötigt
  - Dadurch höhere Verfügbarkeit und geringere Kosten entstehen

Härten des Betriebssystems



- Aktuelle Betriebssystem-Patches einspielen
- Nicht benötigte bzw. unsichere Dienste deaktivieren (R*-Dienste, FTP, Telnet, …)
- Beispiele und Demoanwendungen löschen
- Nicht benötigte Benutzer und nicht benötigten Code entfernen
- Sichere Passworte für Betriebssystem-Benutzer wählen
- Niemals xhost+ verwenden

Anwendungs-Architektur

Die Installation des Application Servers sollte der Architektur der Anwendung angepasst werden

- **Typische Fragen vor der Installation**
  - **Infrastruktur-Datenbank Ja / Nein**
  - **SSO Ja / Nein**
  - **Upload von Dateien notwendig**
  - **Verwendete Komponenten (Forms/Reports/Discoverer/…)**

Einige Tipps

- Verzicht auf die Infrastruktur-Datenbank vereinfacht Handling, ist aber wegen des Verzichts auf SSO/Portal u.U. weniger sicher (Reports Server Secure Mode)
- Niemals den Upload von Dateien (z.B. via Webdav oder Webutil) auf den iAS erlauben, wenn Forms/Reports verwendet werden
- Minimale Anzahl an Komponenten installieren und verwenden

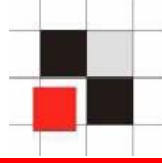
Patching

- Korrektes Patchen des iAS ist die Grundlage eines sicheren Systems
- Details siehe Metalink-Notes 179240.1 [1.0.2.2.x], 215882.1 [9.0.x])
  - Infrastruktur-Datenbank / OID
  - iAS Infrastruktur
  - iAS Home
  - Jinitiator
  - Webutil
  - Security Patches Oracle

Infrastrukturdatenbank

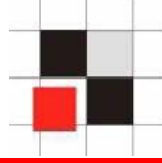
- Default Passworte ändern (Scott, ODS, …)
- TNS Listener sichern
  - TNS_ADMIN_RESTRICTIONS und Listener Passwort setzen
  - IP Restriction verwenden, wenn möglich
  - Extproc entfernen
- PUBLIC-Rechte an mächtigen DB Objekten entziehen (utl_*, dbms_lob, …)

Apache



- Nicht benötigte Module deaktivieren
- Nicht benötigte Apache-Direktiven entfernen
- Administrative URLs via URL-Rewrite schützen
- Log-Files verwenden und regelmäßig auf Probleme kontrollieren
- Apache-Banner entfernen
- Standard Error Seiten ersetzen
- Demo-Programme entfernen

Apache – Config-Dateien bereinigen



- **oracle_apache.conf**
  Nicht benötigte Komponenten deaktivieren
  (z.B. oradav, xml, aq, …)
- **httpd.conf**
  Konfiguration absichern
  (server-status deaktivieren,
  UseWebcacheIP aktivieren,
  ServerSignature & ServerTokens
  modifizieren)
- **mod_oc4j.conf**
  Beispielprogramme deaktivieren
  (z.B. j2ee, repdemo, …)

Webcache

- **Einsatz von Webcache kann zu Problemen mit allow/deny Direktiven führen**

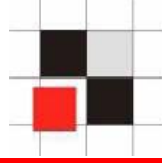
```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from localhost
</Location>
```

- **http://ias/server-status/ ist nicht über Apache direkt zugreifbar, aber über Webcache**
- **Setzen des Wertes UseWebCacheIp On in der httpd.conf löst das Problem**

Absichern von Oracle Forms

- SQL Injection verhindern
- TMP/TEMP/TMP_DIR in ein sicheres Verzeichnis umleiten, da Forms dort z.T. unverschlüsselte Tabellendaten ablegt (iAS Default: \tmp, lesbar für alle)
- Neuste Jinitiator-Version verwenden

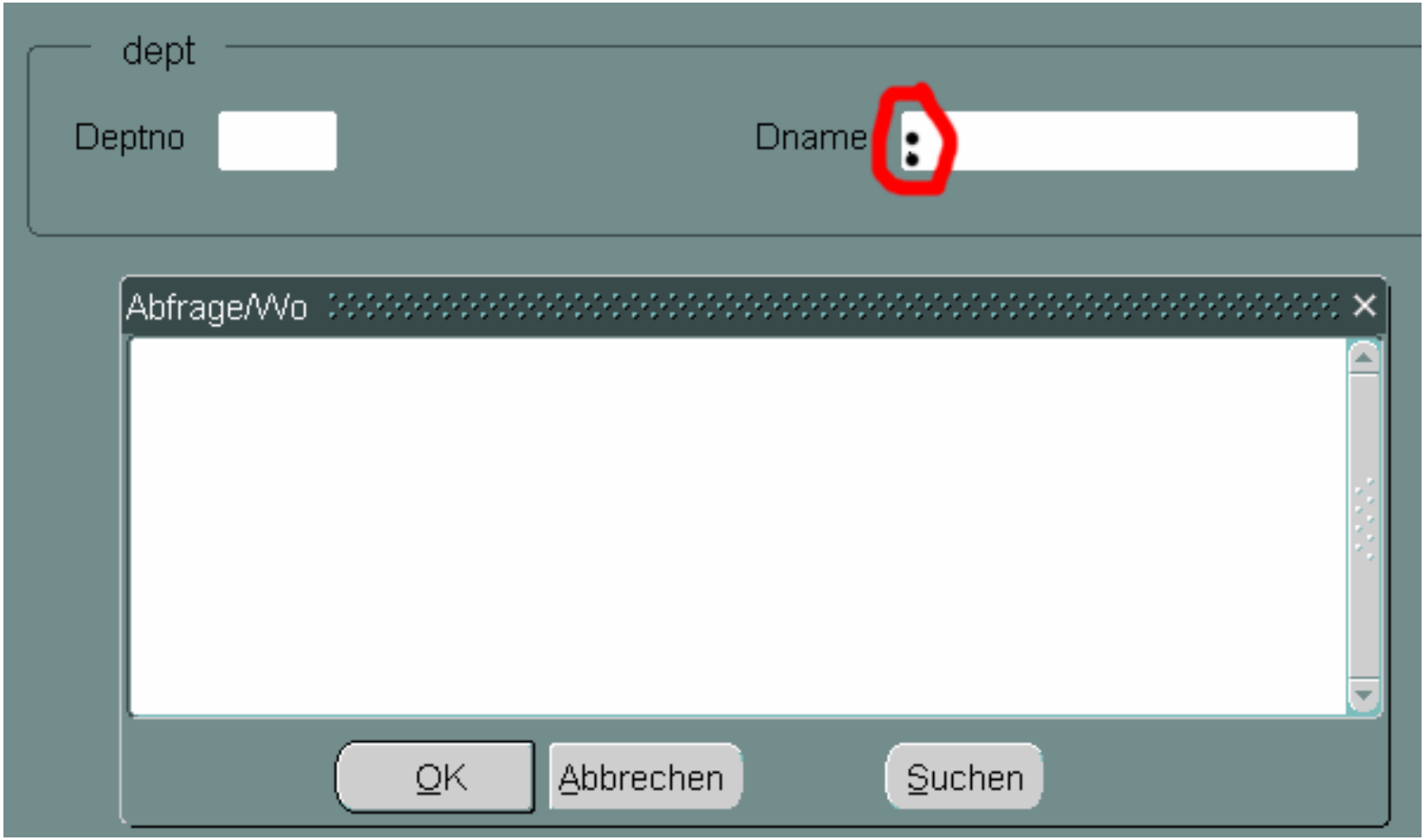
Forms & SQL Injection



- **Enter-Query-Modus erlaubt die Modifikation von Forms-Abfragen**
- **Die speziellen Zeichen :, & und # erlauben es jedem Benutzer, die WHERE-Bedingung zu verändern**
- **Abhängig von der Implementierung innerhalb der Formsanwendung ist die Umgehung des Berechtigungskonzeptes möglich**
- **Transfer von sensiblen Daten ohne Verwendung der Formsmasken nach außen möglich (z.B. via utl_http)**

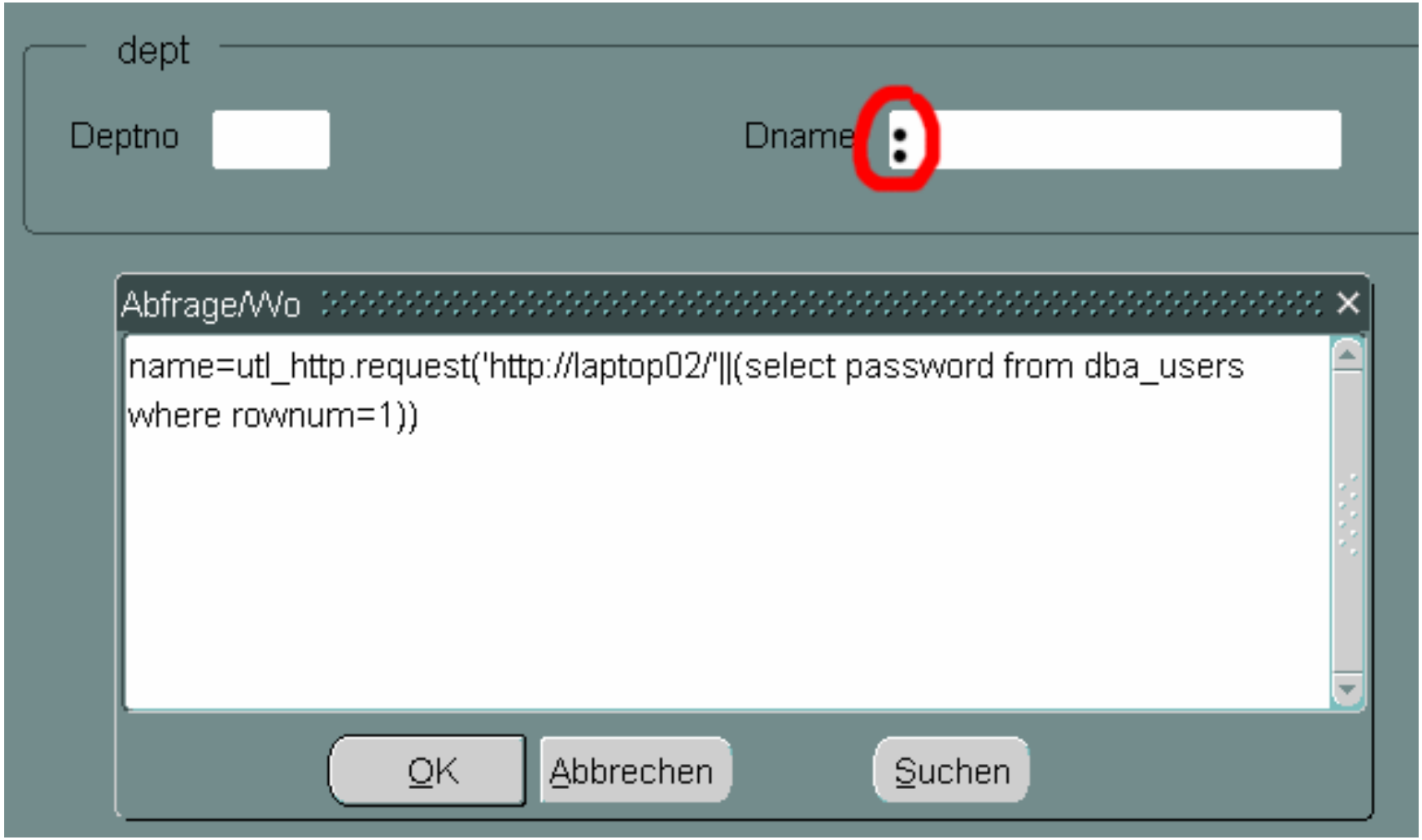
Forms & SQL Injection – Beispiel 1

- **Enter-Query-Modus**

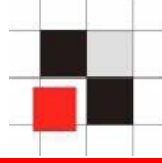


Forms & SQL Injection – Beispiel 2

- **Enter-Query-Modus**



Forms & SQL Injection – Beispiel 3

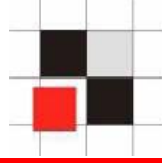


Senden des SYS-Hashkeys zum Webserver des Hackers

Auszug aus dem Apache error_log des Hackers

[Sun Oct 17] [error] [client 192.168.120.254] [ecid: 3093883128448,1] File does not exist: c:/oracle/orafr/apache/apache/htdocs/**af8c688c9aabab74**

Forms & SQL Injection – Lösung 1



- **Setzen der Umgebungsvariablen FORMSxx_RESTRICT_ENTER_QUERY=true**

  (wobei 60 für Forms 6i und 90 für Forms9i/10g verwendet wird)

- **Verhindert die Verwendung der Query/Where-Option**

Forms & SQL Injection – Lösung 2

- **Query/Where via Pre-Query-Trigger deaktivieren** (Metalink Doc.id: 163305.1)
- **Metalink-Note ist jedoch unvollständig, da eine Überprüfung auf & und # fehlt.**

Pre-Query-Trigger

```
:GLOBAL.pre := 'pq';
IF (instr(:dname,':') > 0) OR (instr(:dname,'&') > 0) OR (instr(:dname,'#') > 0)
then
    :dname := Null;
end if;
IF (instr(: deptno,':') > 0) OR (instr(:dname,'&') > 0) OR (instr(:dname,'#') > 0)
then
    :deptno := Null;
end if;
```

Absichern von Oracle Reports

- Verwendeter Reports Modus (mit/oder Portal)
- Absichern der cgicmd.dat
- Environment settings schützen
- Getobjid / showjobs schützen
- Schützen sensibler URLs mit URL-Rewrite
- Google Hacking

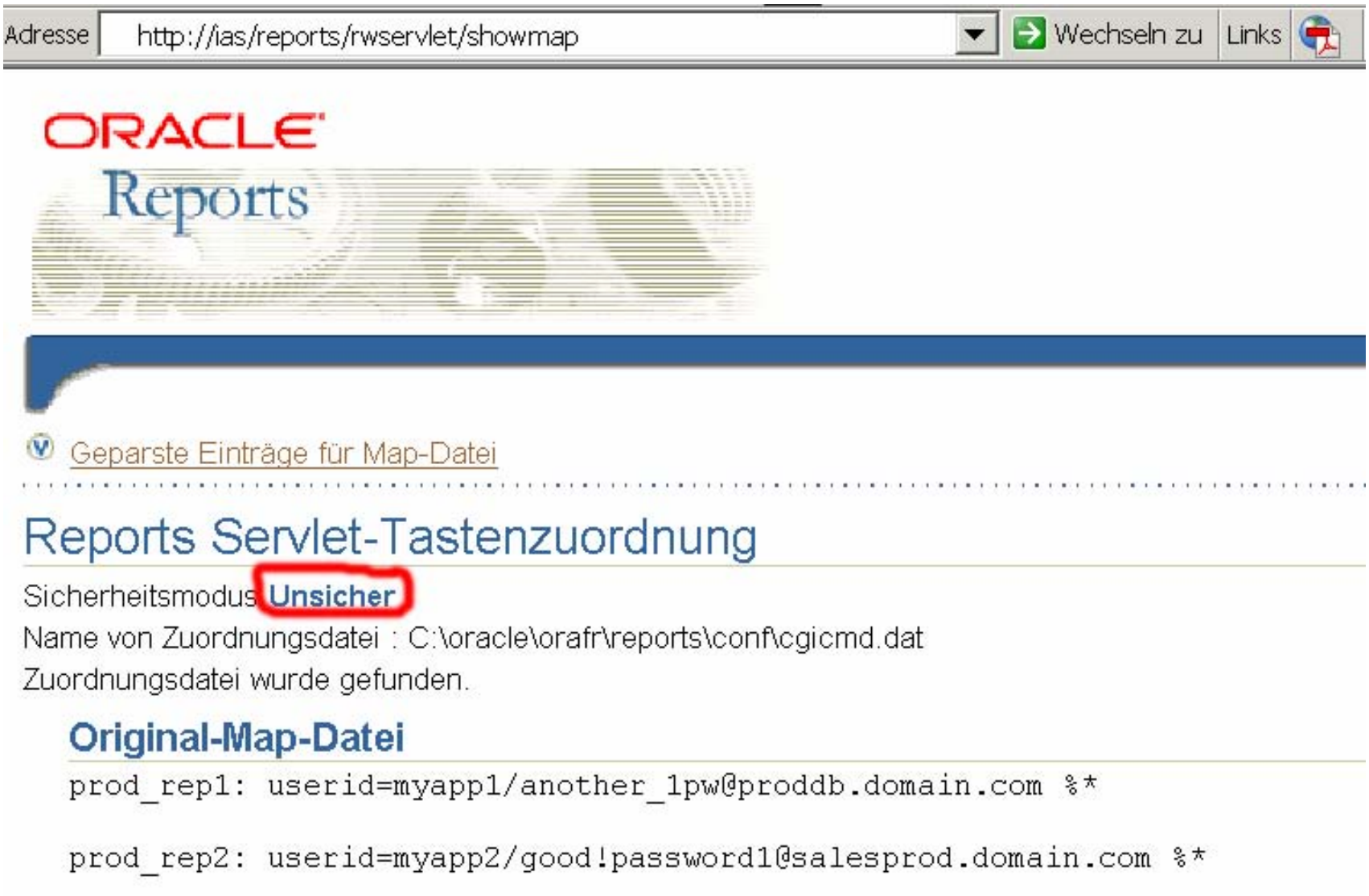
Reports – Sensible URLs

Reports-URLs mit sensiblem Inhalt

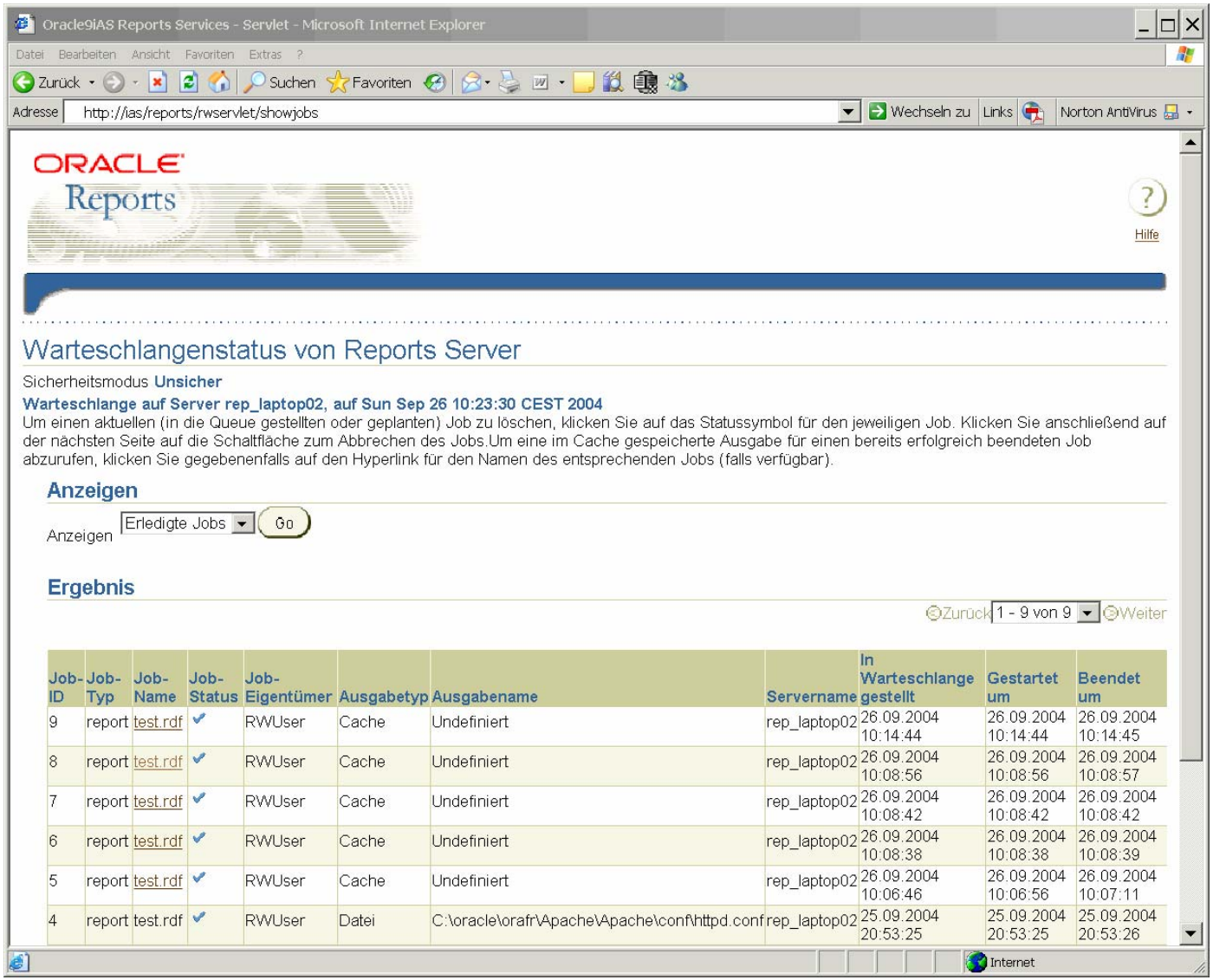
- http://ias/reports/rwservlet/showenv
- http://ias/reports/rwservlet/showmap
- http://ias/reports/rwservlet/showjobs
- http://ias/reports/rwservlet/getjobid7?server=myrep

Reports – Sensible URL - showenv

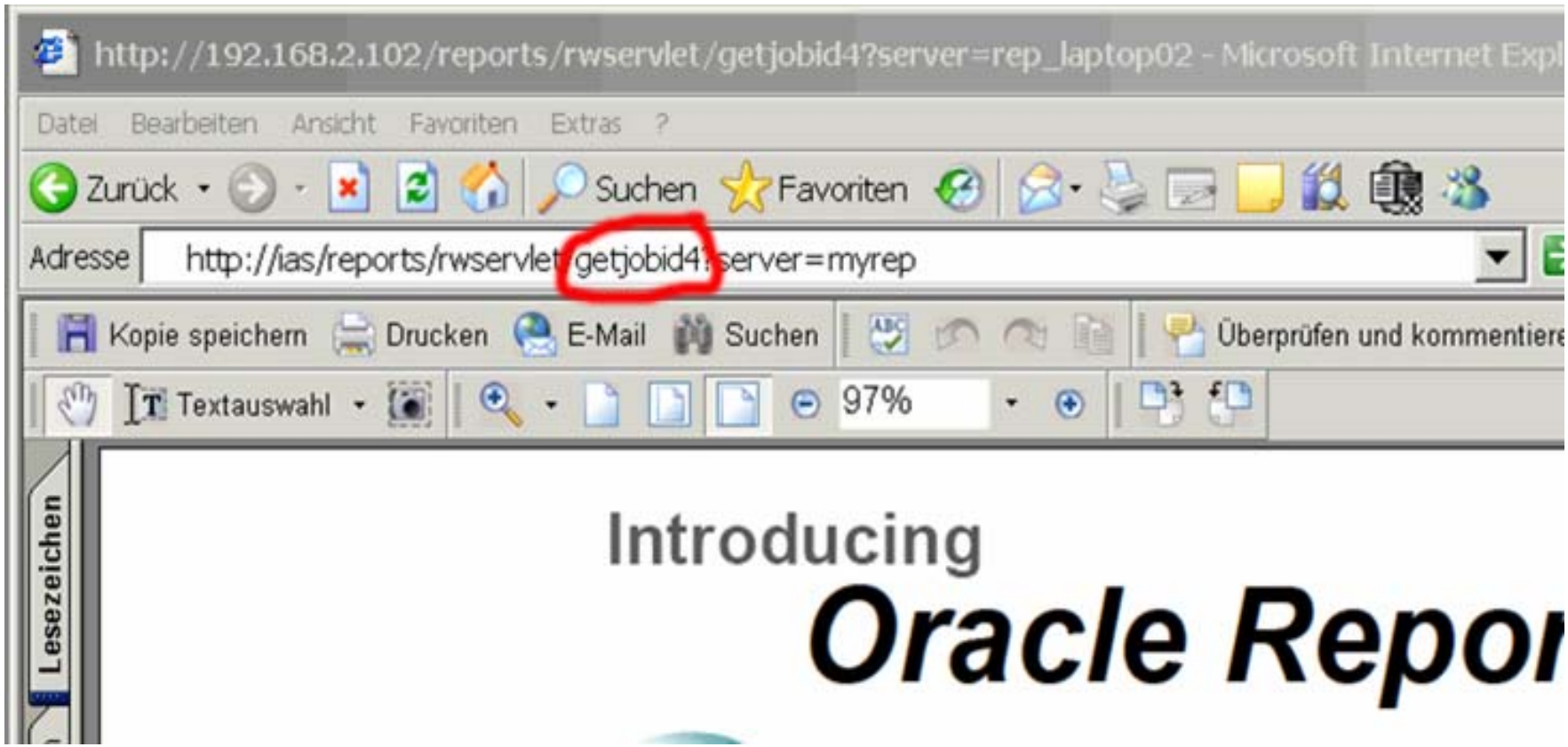
Reports - Sensible URL - showmap



Reports - Sensible URL - showjobs



Reports - Sensible URL - getjobid



Reports – Schutz der sensiblen URLs

- **Schützen der sensiblen URLs**
  - **Setzen der Umgebungsvariablen REPORTSxx_CGINODIAG=No (Testen der Anwendung auf Seiteneffekte xx= 60 für Forms 6i und 90 für 9i/10g)**

und/oder

  - **Blocken der URL mit URL-Rewrite**

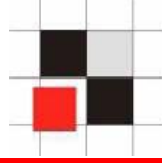
```
RewriteEngine on

RewriteRule ^/reports/rwservlet/showenv(.*)$ /forbidden.htm [R] [NV]
```

Google Hacking – Beispiel 1

Google-Suche nach verwundbaren Reports-Servern

Google Hacking – Beispiel 2



Sensible Inhalte anzeigen, falls der Default-Reportsserver gesetzt ist.

- **Environment zeigen**

http://server/reports/rwservlet/showenv

- **cgicmd.dat anzeigen**

http://server/reports/rwservlet/showmap

- **Jobs anzeigen**

http://server/reports/rwservlet/showjobs

Google Hacking – Beispiel 3

Falls kein Default-Server gesetzt ist, lässt sich dieser sehr einfach herausfinden

- **Paramform an einen bestehenden Report anhängen**

  http://server/reports/rwservlet?business.rdf+2004+0+HTML+cache+paramform=yes

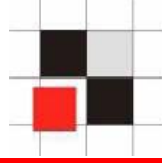
- **Reportsserver generiert eine HTML-Parameterform-Seite, deren HTML-Source folgende Zeile enthält**

  <base href="http://server/servlet/RWServlet/getfile/**rep90_srvr2**/187/35152194.htm">

- **cgicmd.dat anzeigen klappt nun**

http://server/servlet/RWServlet/showmap?server=rep90_srvr2

Weiterführende Links



- **Oracle Security Alerts**
  http://www.oracle.com/technology/deploy/security/alerts.htm

- **Liste mit vielen Dokumenten zum Thema Oracle Security (DB, iAS & Entwicklung)**
  http://www.petefinnigan.com/orasec.htm

- **SANS Step-by-Step Guides**
  http://www.sans.com

red database security

Kontaktadresse:

Red-Database-Security GmbH
Bliesstraße 16
66538 Neunkirchen

Telefon: +49 (0)6821 – 95 17 637
Fax: +49 (0)6821 – 91 27 354
E-Mail: info@red-database-security.com