Hacking & Defending Databases

Todd DeSantis
Technical Pre-Sales Consultant
todd@sentrigo.com

Why Protect The Database?

- Databases hold sensitive information – and lots of it:
  - Customer data, accounts, transactions, payroll, investor data
- When a breach occurs, damage is significant:
  - Direct damages and costs
  - Bad publicity
  - Regulatory penalties

sentrigo™

Know Your Enemy

- Unauthorized access - not just hackers
  - Too many privileges
- Internal attacks
  - Disgruntled employees
  - Just trying to get the job done
  - Industrial espionage, Identity theft, etc.
  - Look around you!!!
- External attacks

sentrigo™

Know Your Enemy

- Hackers are trying:
  - To cause damage
  - Steal
  - Gain access to host systems
- Think like a hacker
  - Learn exploits
  - Look for security issues
    - Configuration, permissions, bugs

sentrigo™

The Problems

- Does a hacker need DBA access?
- Myriads of privileges
  - System level, Application level, Data access
  - Any privelege in the right circumstances can be an issue
- Other issues
  - Incorrect configuration
  - Too many features – large attack surface

sentrigo™

Available Exploits

- Have someone grant you DBA or ALL PRIVILEGES or ALTER USER
- Default passwords
- Password hashes
- Vulnerable code
- Built-in package exploits
  - dbms_metadata.get_ddl
  - ctxsys.driload.validate_stmt
  - Many more

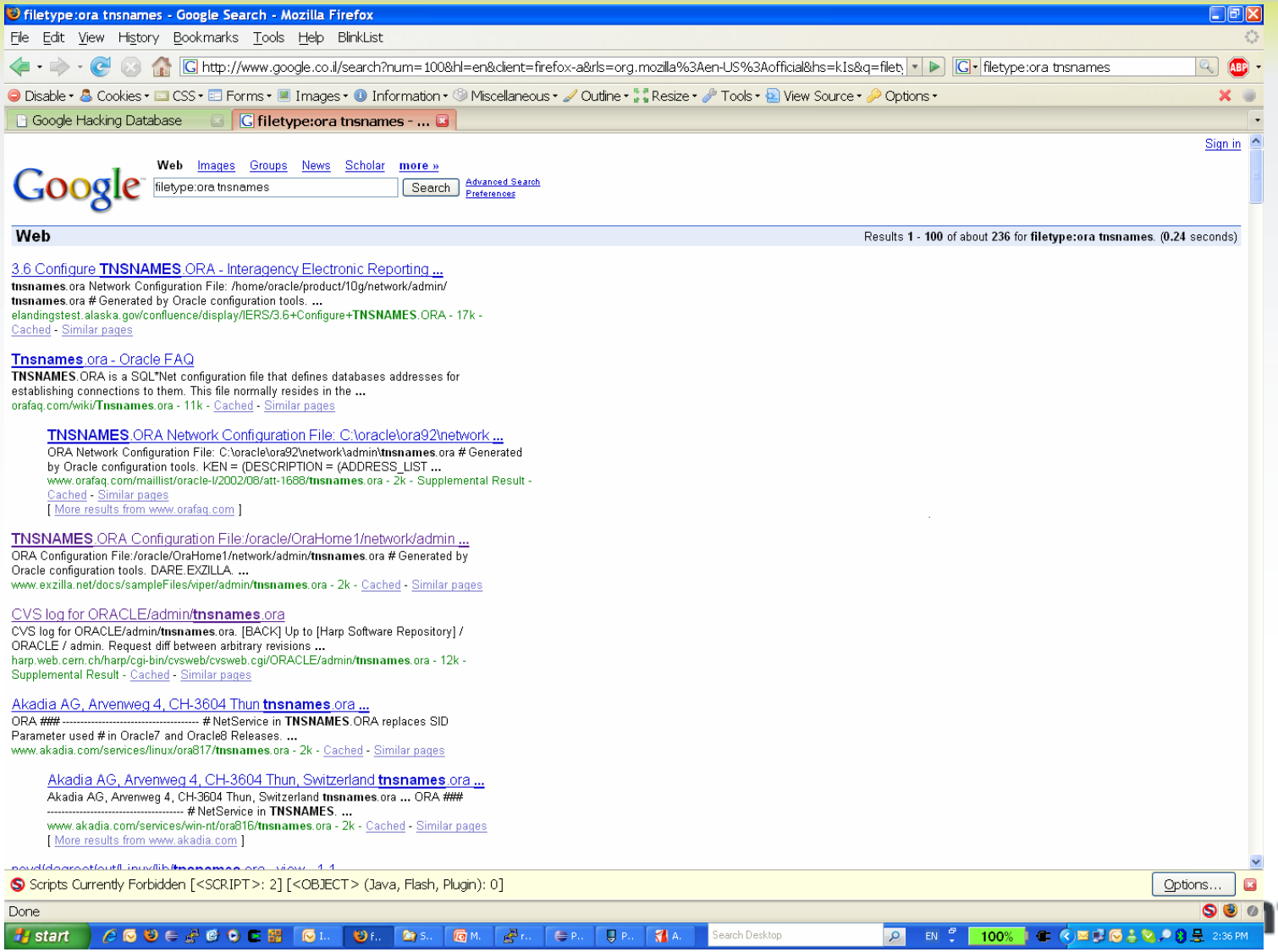
sentrigo™

Finding Available Services

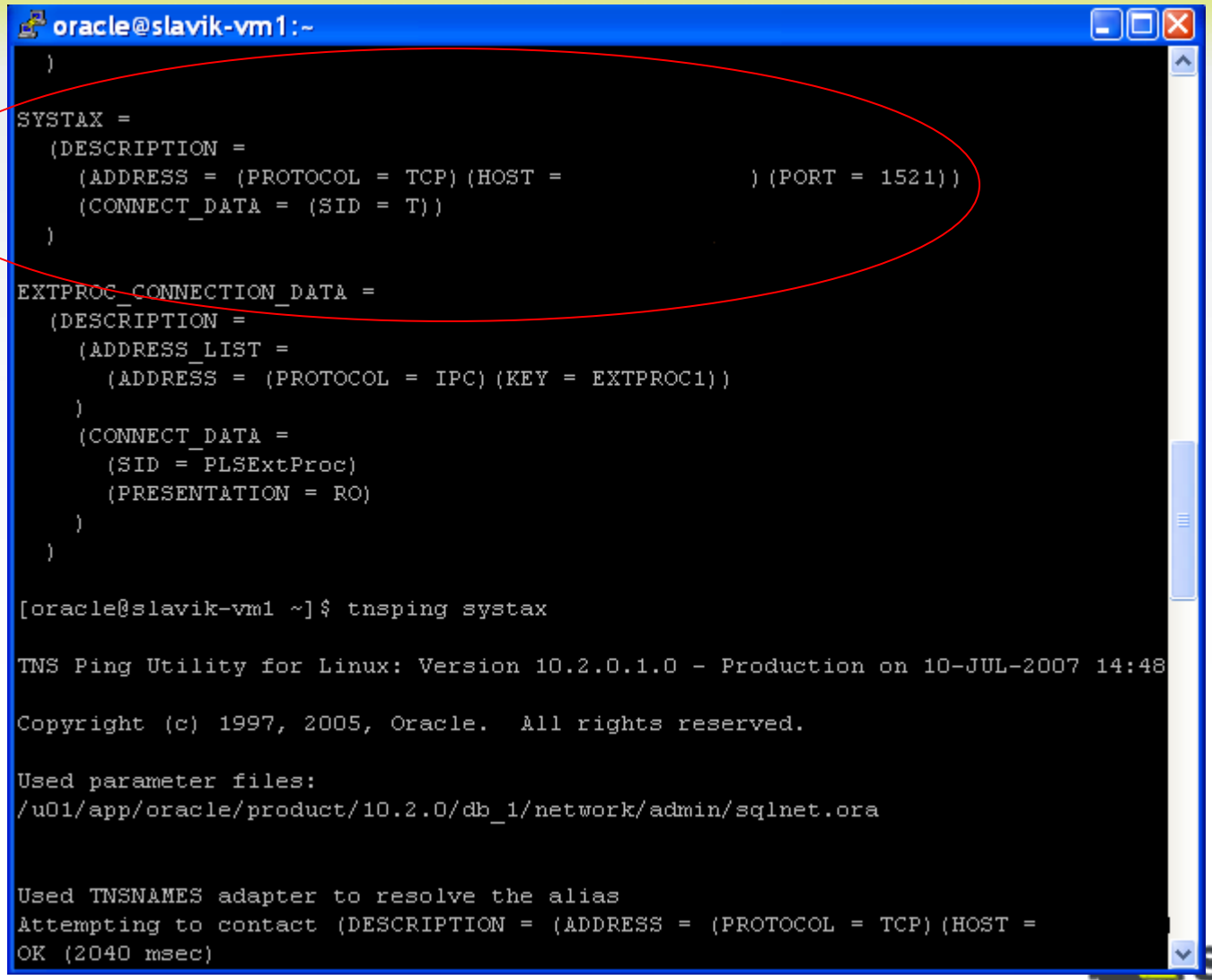
- Google Hacking
  - http://johnny.ihackstuff.com/ghdb.php
  - ora tnsnames
  - iSQL isqlplus
- Use tools for:
  - Brute force password cracking
  - Guessing service names and versions
  - http://www.petefinnigan.com/tools.htm

sentrigo™

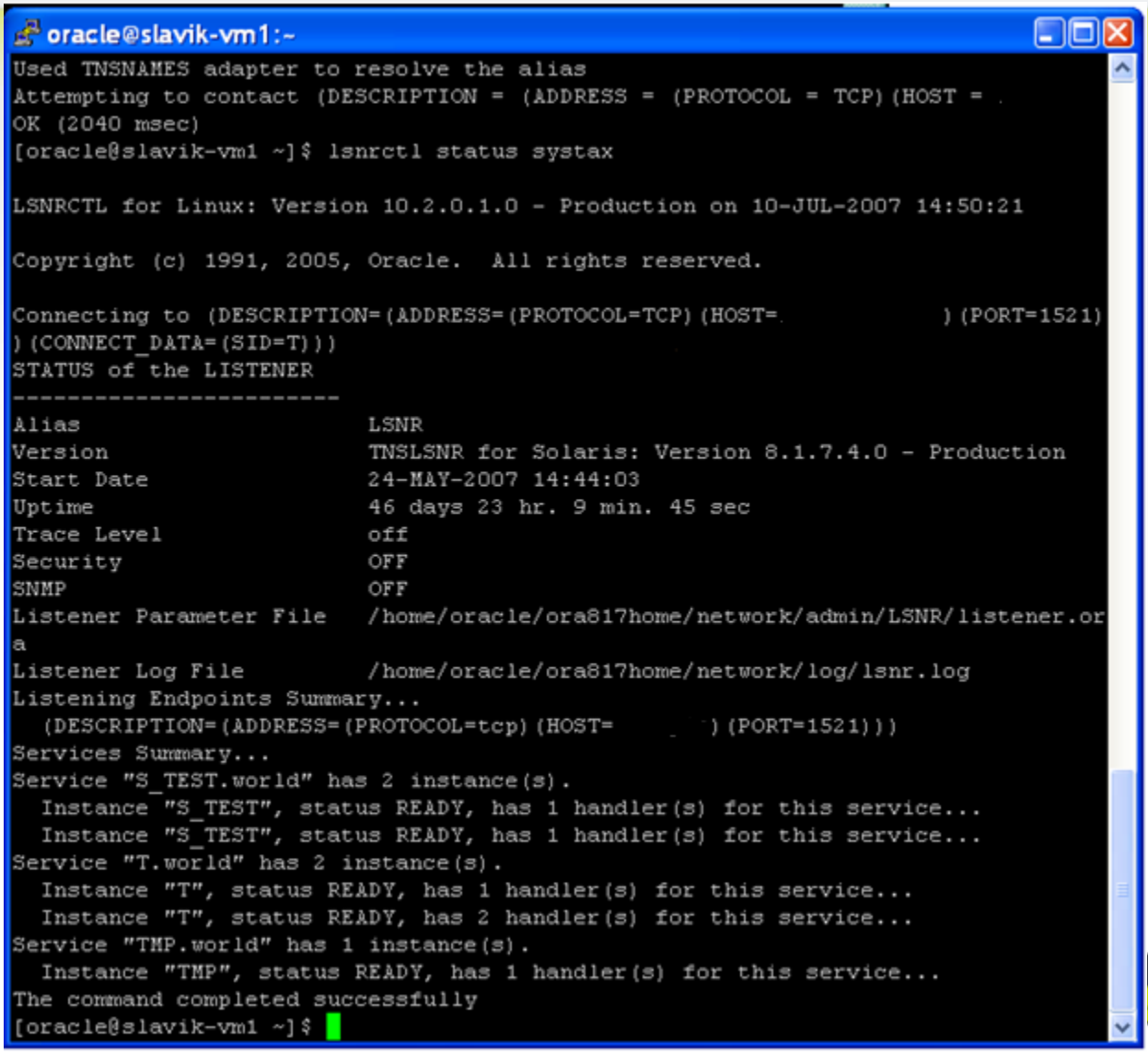
Google Hacking



Google Hacking



Google Hacking



SQL Injection

- Wikipedia –
  - is a technique that exploits a security vulnerability occurring in the database layer of an application. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed.

sentrigo™

SQL Injection

- Exists in
  - Applications
  - Stored program units
    - Build in
    - User created
- Several types
  - Inject SQL, Inject Functions
  - Annonymous blocks of code

sentrigo™

SQL Injection – Web Application

- Username = ' or 1=1 --

  The original statement looked like:

  'select * from users where username = '" + username + '" and password = '" + password + '"

  The result =

  select * from users where username = '' or 1=1 --' and password = ''

sentrigo™

Protecting Your Database

- Apply patch sets, upgrades and CPUs
  - Easier said than done
- Check for default and weak passwords regularly
- Secure the network
  - Listener passwords
  - Valid node checking + firewall
  - Use encryption

sentrigo™

Protecting Your Database

- Install only what you use, remove all else
  - Reduce your attack vector
- The least privilege principle
  - Lock down packages
    - System access, file access, network access
- Encrypt critical data
- Use secure coding techniques
  - Bind variables, ownership

sentrigo™

Protecting Your Database

- Try out the Hedgehog - http://www.sentrigo.com
  - Virtual patching
  - SQL Injection protection
  - Fine grain auditing
  - Centralized management
  - Terminate rogue sessions
  - More

sentrigo™

Questions?

sentrigo™