# The Untold Tale of Database Communication Protocol Vulnerabilities

**Amichai Shulman, CTO**

**BlackHat 2007 – Washington DC**

# Agenda

- A Brief History of Database Security Threats
- Introduction to Database Communication Protocols (and their problems)
- Detailed Walk Through of Vulnerabilities
- Mitigation Techniques

# A Brief History of Database Security Threats

- Infrastructure Attacks
  - Targets generic network stack vulnerabilities or common services on a server
  - Unrelated to the role of the server as a DB server
  - Not DB vendor specific
  - Proactive mitigation using network FW
  - Reactive mitigation using IDS / IPS

# A Brief History of Database Security Threats

- Privilege Abuse Using SQL Queries
  - Compromised credentials
  - Excessive privileges
  - Inherent to database servers
  - Not DB vendor specific
  - Proactive mitigation using internal access controls

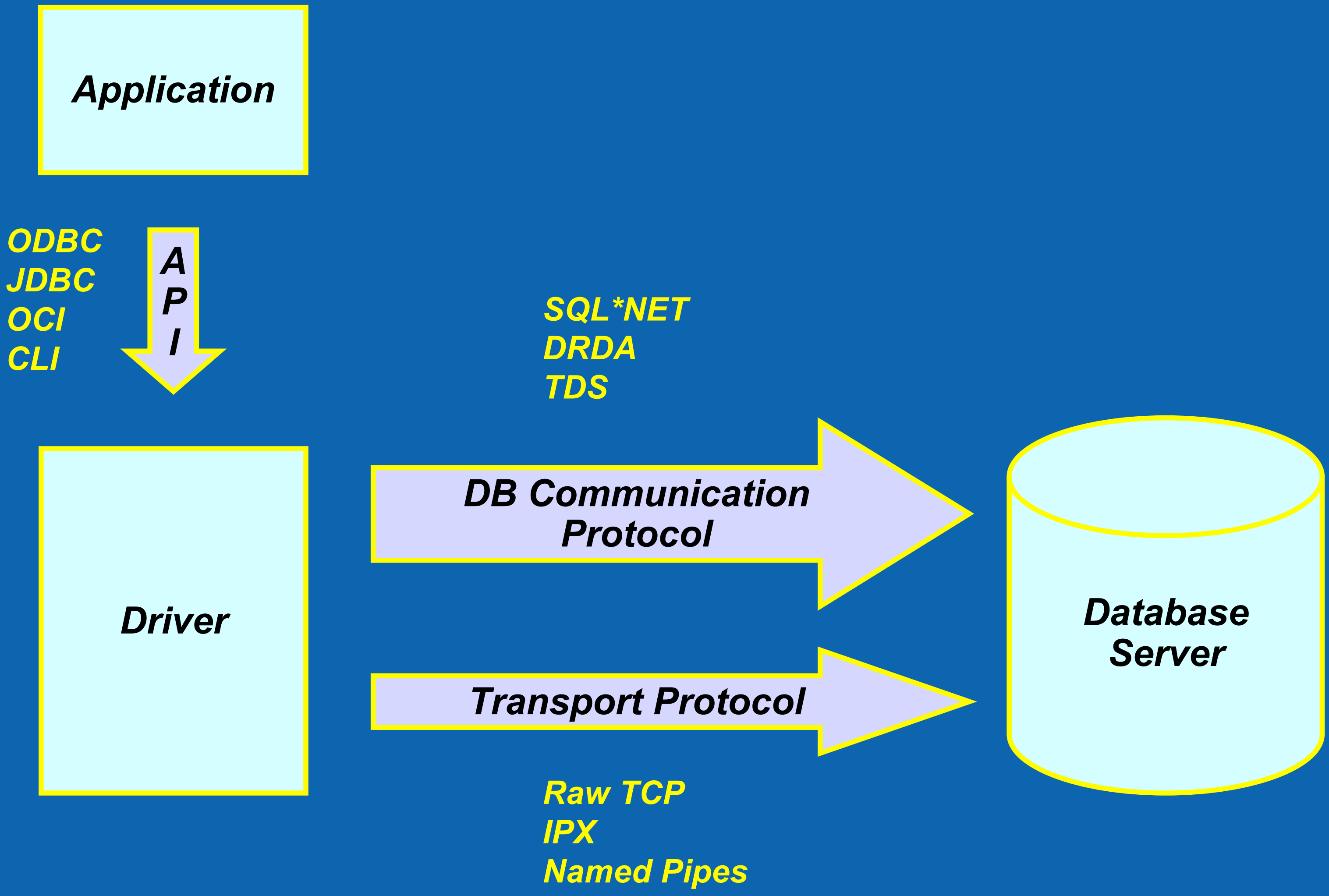
# A Brief History of Database Security Threats

- SQL Level Vulnerabilities
  - Buffer overflow
    - xp_SetSQLSecurity, xp_sprintf, pwdencrypt (MS SQL)
    - Pwdencrypt (MS SQL)
    - CREATE DATABASE LINK (Oracle)
  - SQL Injection
    - driload.validate_stmt, dbms_metadata.get_ddl (Oracle)
  - Privilege elevation
    - OpenRowset (MS SQL)
    - Modify Data via Inline View (Oracle)
- Vendor Specific
- Proactive Mitigation
  - DB configuration
  - Access control
- Reactive Mitigation
  - Patching

# A Brief History of Database Security Threats

- Database Communication Protocol Vulnerabilities
  - First peeks on 2000
  - A major surge during 2006
    - Oracle (~20)
    - DB2 (~10)
    - Informix (~10)
    - MS SQL (<5)
  - Database vendor specific

# Database Communication Protocols
# Introduction



## *Database Communication Protocols*
# Introduction

- SQL is standard however,
- No standard exists for the following tasks:
  - Creating client session
  - Conveying commands from client to server
  - Conveying data and status from server to client
  - Implementing cursor command and prepared statements
- Vendors are filling the gap with proprietary technology:
  - Messages
  - Sequences
  - Semantics

# *Database Communication Protocols*
## Introduction

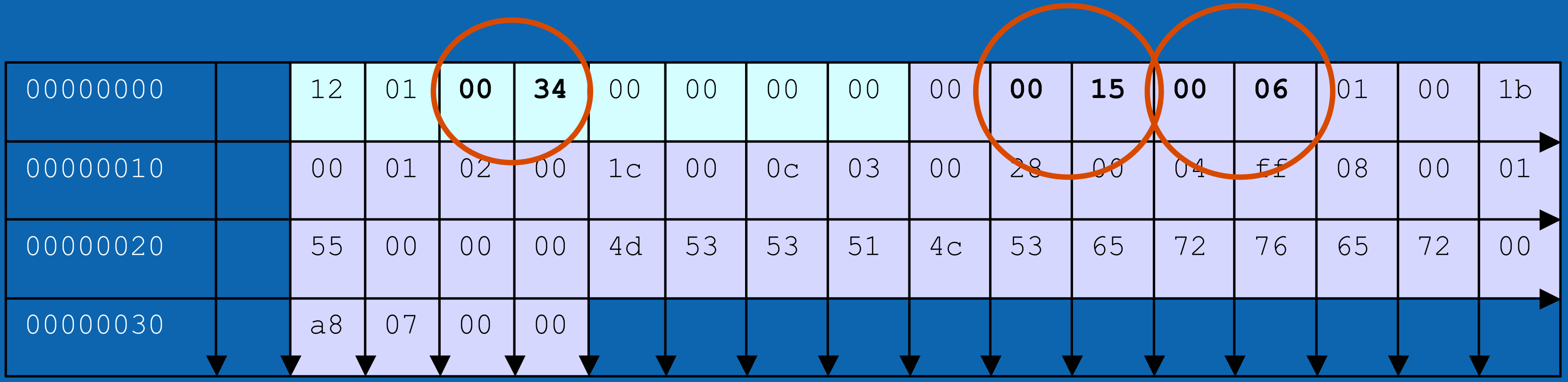
| | |
|---|---|
| Oracle | SQL*NET (AKA Net8, Net9) |
| IBM | DRDA (replacement of DB2RA) |
| Sybase | TDS 5 (extending TDS 4.2) |
| MS SQL | TDS 7 & 8 (extending TDS 4.2) |

# *Database Communication Protocols*
# Maximum Complexity

- Multiple layers
  - TDS: 2 layers (~10, ~100)
  - DRDA, SQL*NET: 3 layers
  - Sometimes there is redundancy between layers (size fields, offsets, termination tokens)
  - Each layer is handled independently

## *Database Communication Protocols*
# Maximum Complexity

- Microsoft TDS
- Hello Message



```
00000000   12 01 00 34 00 00 00 00 00 00 15 00 06 01 00 1b
00000010   00 01 02 00 1c 00 0c 03 00 28 00 04 ff 08 00 01
00000020   55 00 00 00 4d 53 53 51 4c 53 65 72 76 65 72 00
00000030   a8 07 00 00
```

# *Database Communication Protocols*
# Maximum Complexity

- Long history of backwards compatibility
  - Oracle 8 through Oracle 10g
  - TDS 4.2 through TDS 9
  - TDS 5 duplicate set of commands
- Data representation
  - Try to bridge different client and server environments
  - Fixed for lower layer, negotiable for other layers (Endianess, String representation, etc.)
  - DRDA uses 8 different code pages for protocol messages
  - Oracle has 3 different data representations
  - Oracle eliminates multiple network transmissions of identical values

# *Database Communication Protocols*
# Minimum Scrutiny

- Vendors are (almost) exclusive producers of basic client software
  - Some exceptions like DataDirect's drivers and FreeTDS implement subsets of the protocols
- Server side protocol implementation is not tested against spec but against client implementation
  - Driver developers are not always aware of the full capabilities of the protocol
- Probably few out-of-spec testing.
  - Especially for backwards compatible code
- Spec is not open for public review
  - DRDA is an exception

## *Database Communication Protocols*
# Bring in the Vulnerabilities!

- Analysis of protocols is required for network based database security gateways
- Simple analysis can be done using a network sniffer

***DEMO***

# *Database Communication Protocols*
# Bring in the Vulnerabilities!

- Vulnerability research of connection establishment can be done using simple tools like *netcat*
- Deeper analysis of the protocol and vulnerability research into other parts of it requires a different tool
  - Changing selectively parts of messages within an existing connection
  - Injecting messages into existing connection
  - Removing messages from a connection
- Introducing:

***TCPirate***

# *Database Communication Protocols*
## TCPirate

- Interactive TCP Proxy
- Record messages in both directions
- Messages can be trapped
  - Inspect message
  - Make controlled changes to the message before letting it go
  - Replace the message with a message from a file
  - Drop the message
- Spontaneously inject messages into a connection

***DEMO***

## *Vulnerability Details*
# Classification

- Message Structure Tampering
- Field Size Tampering
- Field Content Manipulation
- Message Sequence Tampering

# *Vulnerability Details*
# Message Structure Tampering

- Message fields are explicitly declared (e.g. DRDA, Some Oracle Messages)
- Three main techniques
  - Removing fields from a message
  - Adding fields to a message or duplicating fields in a message
  - Combining fields in an unexpected manner

## *Vulnerability Details*

# Message Structure Tampering

- BID 19586, Denial of Service vulnerability patched by IBM
- Fields in DRDA messages are explicitly declared
- RDBNAM field (code 0x2110) can be omitted from connection request message
- Server becomes unstable upon connection

# *Vulnerability Details*
# Message Structure Tampering

- Original Message

```
00000000    00  b4  d0  41  00  01  00  ae  10  41  00  6e  11  5e  84  82
00000010    f2  82  97  4b  85  a7  85  40  40  40  40  40  40  40  40  40
00000020    40  40  f0  f5  c6  f8  f0  f5  f5  f4  f0  f0  f0  e6  00  c5
00000030    00  c2  00  c3  00  d6  00  c8  00  d6  00  d9  00  e3  00  00
00000040    00  00  00  00  00  00  00  00  00  00  00  00  00  60  f0  f0
00000050    f0  f1  c1  d4  c9  c3  c8  c1  c9  40  40  40  40  40  40  40
00000060    40  40  40  40  40  40  40  40  40  40  40  40  40  40  40  40
00000070    c3  e3  d6  c4  c2  40  40  40  00  18  14  04  14  03  00  07
00000080    24  07  00  07  14  74  00  05  24  0f  00  07  14  40  00  07
00000090    00  0b  11  47  d8  c4  c2  f2  61  d5  e3  00  0d  11  6d  c5
000000A0    c4  e5  c9  c3  c5  60  c1  d4  00  0c  11  5a  e2  d8  d3  f0
000000B0    f8  f0  f1  f5  00  4a  d0  01  00  02  00  44  10  6d  00  06
000000C0    11  a2  00  09  00  16  21  10  e2  c1  d4  d7  d3  c5  40  40
000000D0    40  40  40  40  40  40  40  40  40  40  00  24  11  dc  5c  17
000000E0    36  09  dd  e8  92  88  f4  e3  79  b0  57  9d  05  36  e1  26
000000F0    f6  ce  a9  90  e7  8d  86  09  e8  36  d0  95  e0  32
```

## *Vulnerability Details*

# Message Structure Tampering

- Tampered Message

```
00000000  00  b4  d0  41  00  01  00  ae  10  41  00  6e  11  5e  84  82
00000010  f2  82  97  4b  85  a7  85  40  40  40  40  40  40  40  40  40
00000020  40  40  f0  f5  c6  f8  f0  f5  f5  f4  f0  f0  f0  e6  00  c5
00000030  00  c2  00  c3  00  d6  00  c8  00  d6  00  d9  00  e3  00  00
00000040  00  00  00  00  00  00  00  00  00  00  00  00  00  60  f0  f0
00000050  f0  f1  c1  d4  c9  c3  c8  c1  c9  40  40  40  40  40  40  40
00000060  40  40  40  40  40  40  40  40  40  40  40  40  40  40  40  40
00000070  c3  e3  d6  c4  c2  40  40  40  00  18  14  04  14  03  00  07
00000080  24  07  00  07  14  74  00  05  24  0f  00  07  14  40  00  07
00000090  00  0b  11  47  d8  c4  c2  f2  61  d5  e3  00  0d  11  6d  c5
000000A0  c4  e5  c9  c3  c5  60  c1  d4  00  0c  11  5a  e2  d8  d3  f0
000000B0  f8  f0  f1  f5  00  34  d0  01  00  02  00  2e  10  6d  00  06
000000C0  11  a2  00  09  00  24  11  Dc  5c  17  36  09  Dd  E8  92  88
000000D0  F4  E3  79  B0  57  9d  05  36  E1  26  F6  Ce  A9  90  E7  8d
000000E0  86  09  E8  36  D0  95  E0  32
000000F0
```

## *Vulnerability Details*

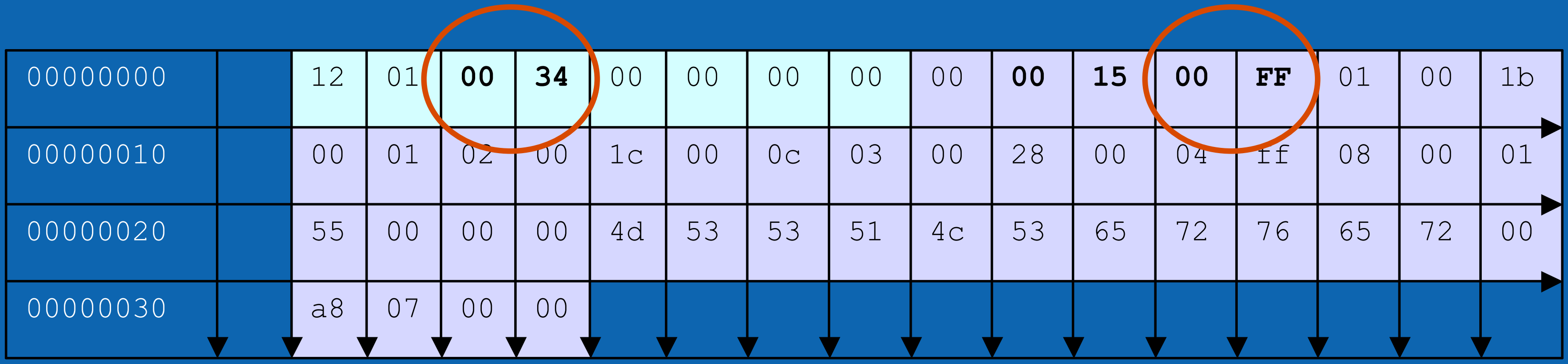
# Field Size Manipulation

- Field size is explicitly declared using another dedicated field
- Mostly used for buffer overflow attacks
  - The length indicator field is capable of expressing larger data sizes than actually supported by server
- Example 1:
  - BID 18428, Buffer overflow vulnerability in DB2 connection request
  - A field called MGRLVLLS is extended to include more than 400 bytes
  - Unauthenticated denial of service and possible execution of arbitrary code
  - Affects all platforms including OS/390!

## *Vulnerability Details*

# Field Size Manipulation

- Example 2:
  - MSDE, Hello message
  - Abuse redundancy of size information
  - Dump internal buffers



```
00000000    12 01 00 34 00 00 00 00 00 00 15 00 FF 01 00 1b
00000010    00 01 02 00 1c 00 0c 03 00 28 00 04 ff 08 00 01
00000020    55 00 00 00 4d 53 53 51 4c 53 65 72 76 65 72 00
00000030    a8 07 00 00
```

**DEMO**

## *Vulnerability Details*
# Field Content Manipulation

- Worst type of vulnerabilities
- Example 1:
  - US CERT Vulnerability Note VU#871756
  - Oracle TNS protocol fails to properly validate authentication requests
  - One of the login messages contains an SQL query that is executed under the SYS security context
  - The query is presumable hard-coded in the driver software
  - Can be exploited by simple editing of client side DLL
  - Affects all Oracle versions from 8 to 10gR2

***DEMO***

# *Vulnerability Details*
# Field Content Manipulation

- Example 2:
  - MS SQL Server trace evasion
  - Driver does not allow for account name in login message to contain non-printable ASCII characters
  - Construct a login message that includes a valid account name preceded by NULL character
  - Authentication mechanism disregards the extra character
  - Trace mechanism tries to process it
  - Consequence: Invisible users

***DEMO***

# *Vulnerability Details*
# Field Content Manipulation

- Manipulate the sequence of messages within the connection in an unexpected manner
- Two example
  - Oracle
  - Informix
- Details cannot be disclosed as far as vulnerabilities are not patched
- Known effects
  - Unauthenticated access to server
  - Denial of service

# Mitigation

- Internal server control are useless
  - They are the ones with the vulnerability
- Patching
  - Simply not fast enough
- Reactive mitigation through IPS / IDS
  - Some vulnerabilities can be addressed using signatures or Snort-like rules

# Mitigation

- Proactive mitigation with Database Security Gateway
  - Network device aware of the database communication protocol
  - Parses the data stream
  - Alert on messages that do not conform with expected client behavior
  - “Expected Client Behavior” is defined through research of vendor supplied drivers.

# Question & Answer

# Thank You

## Imperva Inc.

950 Tower Lane, Suite 1550
Foster City, CA 94404
Sales: (866) 926-4678

www.imperva.com