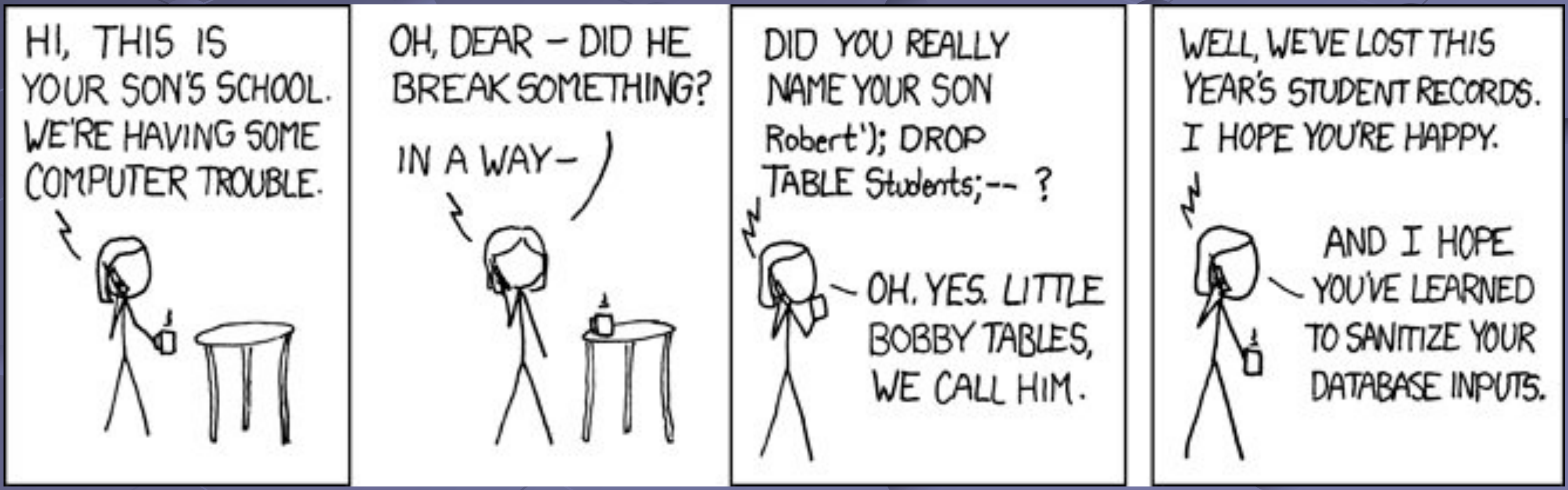
Database System Security

Paul J. Wagner

UMSSIA 2008


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Need for Database System Security Education

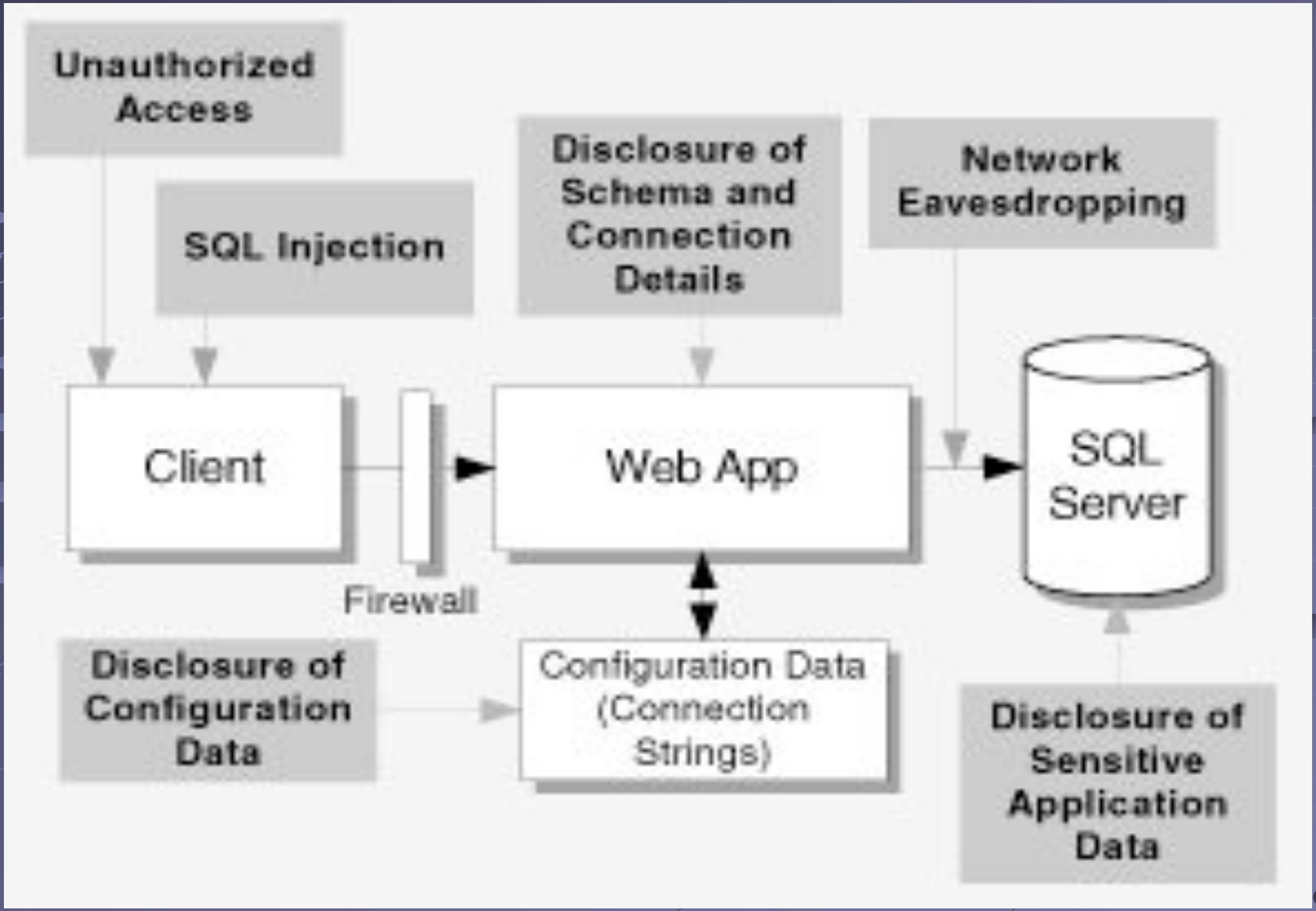
- "The value is in the data…" – 3M Poster
- Attacks have changed from glory-seeking to attempted financial gain
- Security curriculum is relatively light in database systems area
  - Focus currently on protecting information through network configuration, systems administration, application security

Goals

- Understand security issues in a general database system environment, with examples from specific database management systems (DBMSs)
- Consider database security issues in context of general security principles and ideas
- Examine issues relating to both database storage and database system communication with other applications

Main Message

- Database system security is more than securing the database; to achieve a secure database system, we need a:
  - Secure database
  - Secure DBMS
  - Secure applications / application development
  - Secure operating system in relation to database system
  - Secure web server in relation to database system
  - Secure network environment in relation to database system



Unauthorized Access
SQL Injection
Disclosure of Schema and Connection Details
Network Eavesdropping
Client
Firewall
Web App
SQL Server
Disclosure of Configuration Data
Configuration Data (Connection Strings)
Disclosure of Sensitive Application Data

diagram from .NET Framework Security article, Meier et. al.

Secure Databases

- Database – a domain-specific collection of data; e.g. UWEC student database
- Historical database security framework
  - Separate set of usernames, passwords
  - Database administrator assigns privileges for each user

Secure Databases (2)

- Security issues
  - Default users/passwords, especially if have higher-level privileges
    - Oracle: sys, system accounts – privileged (Oracle 8i and prior - with default passwords)
    - Oracle: scott account – well-known account and password, part of public group
      - e.g. public can access all_users table
  - General password policies (length, domain, changing, protection)

Secure Databases (3)

- Managing Privileges
  - Privilege types
    - System - actions
    - Object – data
  - Problem: M (users) x N (privileges) possible combinations
  - Solution: Roles (a security pattern)
    - Collections of (usually system) privileges
  - Manage with Grant / Revoke, higher level tools
    - Giving (or removing) privileges or roles to (from) users

Secure Databases (4)

- Encryption of Data
  - Sensitive data should be encrypted within database
  - Most DBMSs now have utilities for this
  - Examine cost of this process
    - May slow access time, use resources
    - May interfere with system administration
  - Don’t forget about data once out of the database
    - Need to decrypt for users
    - Ensure data is still protected when DB sends it to a client for usage, or when client sends to DB

Secure Databases (5)

- Other Database Issues – Inference Attack
  - Statistical Database – database used for statistical purposes, usually Online Analytical Processing (OLAP) as in data warehouse sitituations
    - Generally allows only aggregate function queries (may be limited to SUM, COUNT, AVERAGE) on a group of rows
    - May not allow queries on individual rows
  - Problem: it may still be possible to *infer* individual data values from such a database

Secure Databases (6)

- Inference Attack Example:
  - Base query: find count of all UWEC students
  - Example SQL:
    - SELECT Count(*) FROM StudentDB;
  - returns single number, ~10,000
  - doesn’t expose any individual information, but we can do more…

Secure Databases (7)

- Inference Attack Example (2):
  - Find count of UWEC students that:
    - Have a local address in Eau Claire
    - Have a GPA of between 3.1 and 3.2
    - Are majoring in computer science
  - SELECT Count(*) FROM StudentDB WHERE cond1 AND cond2 AND cond3 AND …
  - Getting count down to fewer and fewer students as we add filter conditions
  - If count gets down to 1, you know an individual exists with those characteristics

Secure Databases (8)

- May even be able to gain private information
  - Find average hourly pay rate for UWEC students that have the above characteristics
- Inference Attack Prevention Techniques:
  - Return an approximate number of rows (e.g. between A and B)
  - Limit the number of filtering conditions on a query
  - Return ranges for values, especially for sensitive fields (e.g. salary falls between X and Y)
- Difficult to balance functionality and security with such data

Secure Databases (9)

- Other Database Issues - Correlation Attack
  - Data in database may be anonymous
  - However, it may be able to be correlated with another set of data which is not anonymous
    - Anonymity can thus be removed
  - Example: Statistical analysis of public but anonymized Netflix data set using Internet Movie Database (imdb.com) data
    - By comparing rankings and timestamps from Netflix data, authors identified a subset of individuals through correlated data from IMDB

Secure Databases (10)

- More Information on this attack
  - http://www.schneier.com/blog/archives/2007/12/anonymity_and_t_2.html - Discussion of problem in general, this analysis, other similar efforts
  - http://www.cs.utexas.edu/~shmat/shmat_netflix-prelim.pdf - Paper on the original analysis

Secure DBMS

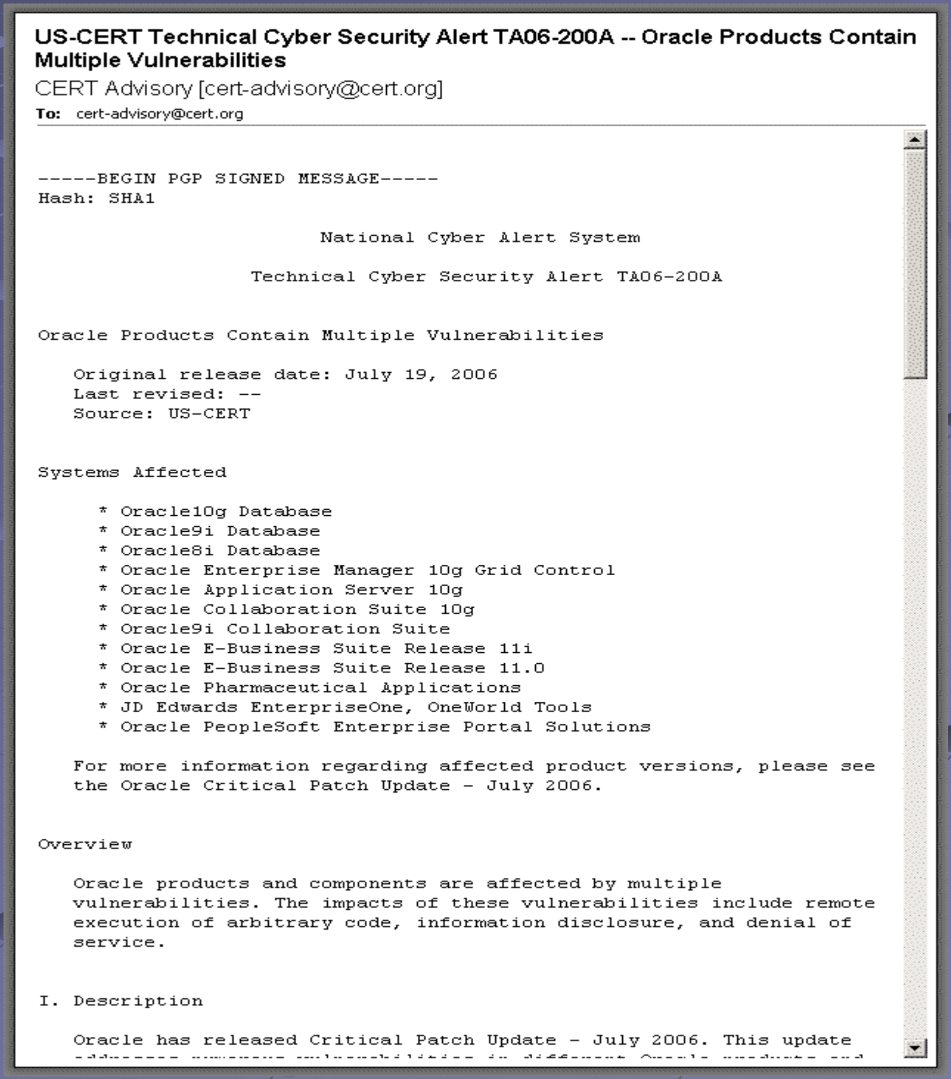
- Database Management System (DBMS) – the domain-independent set of software used to manage and access your database(s)
- Many Possible Holes in DBMS software
  - http://technet.oracle.com/deploy/security/alerts.htm (50+ listed)
  - Majority of problems - buffer overflow problems in (legacy) DBMS code
  - Miscellaneous attacks (Denial of Service, source code disclosure of stored procedures and JSPs, others)
- DBMS is not an island
  - Oracle example - UTL_FILE package in PL/SQL (Oracle's procedural language on top of SQL)
    - allows read/write access to files in directory/ies specified in utl_file_dir parameter in init.ora
    - possible access through symbolic links

Secure DBMS (2)

- Need for continual patching of DBMS
  - Encourage awareness of issues, continuous vigilance as a database administrator
  - Cost of not patching
    - SQL Slammer Worm - ~100,000 systems affected
      - Even though patch had been available for 6 months

Secure DBMS (3)

- US-CERT advisories
  - List of major software packages currently watched includes Oracle



US-CERT Technical Cyber Security Alert TA06-200A -- Oracle Products Contain Multiple Vulnerabilities

CERT Advisory [cert-advisory@cert.org]

To: cert-advisory@cert.org

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

                        National Cyber Alert System

                  Technical Cyber Security Alert TA06-200A


Oracle Products Contain Multiple Vulnerabilities

   Original release date: July 19, 2006
   Last revised: --
   Source: US-CERT


Systems Affected

     * Oracle10g Database
     * Oracle9i Database
     * Oracle8i Database
     * Oracle Enterprise Manager 10g Grid Control
     * Oracle Application Server 10g
     * Oracle Collaboration Suite 10g
     * Oracle9i Collaboration Suite
     * Oracle E-Business Suite Release 11i
     * Oracle E-Business Suite Release 11.0
     * Oracle Pharmaceutical Applications
     * JD Edwards EnterpriseOne, OneWorld Tools
     * Oracle PeopleSoft Enterprise Portal Solutions

   For more information regarding affected product versions, please see
   the Oracle Critical Patch Update - July 2006.


Overview

   Oracle products and components are affected by multiple
   vulnerabilities. The impacts of these vulnerabilities include remote
   execution of arbitrary code, information disclosure, and denial of
   service.


I. Description

   Oracle has released Critical Patch Update - July 2006. This update
```

Secure Application Development

- Access to Oracle Database or Environment Through Applications
- Need to consider security of applications using database as well as security of data in database itself
- Example: SQL Injection Attack

SQL Injection

- SQL Injection
  - Definition – inserting malicious SQL code through an application interface
    - Often through web application, but possible with any interface
  - Typical scenario
    - Three-tier application (web interface, application, database)
    - Overall application tracks own usernames and passwords in database (advantage: can manage users in real time)
    - Web interface accepts username and password, passes these to application layer as parameters

SQL Injection (2)

- Example: Application Java code (receiving username and password from client) contains SQL statement:
  - String query = "SELECT * FROM users_table " + " WHERE username = " + " ‘ " + username + " ‘ " + " AND password = " + " ‘ " + password + " ‘ "
  - Expected Result: "SELECT * FROM users_table WHERE username = ‘wagnerpj‘ AND password = ‘paulpass‘;
  - Note: String values must be single quoted in SQL, so application provides these before and after each passed string parameter

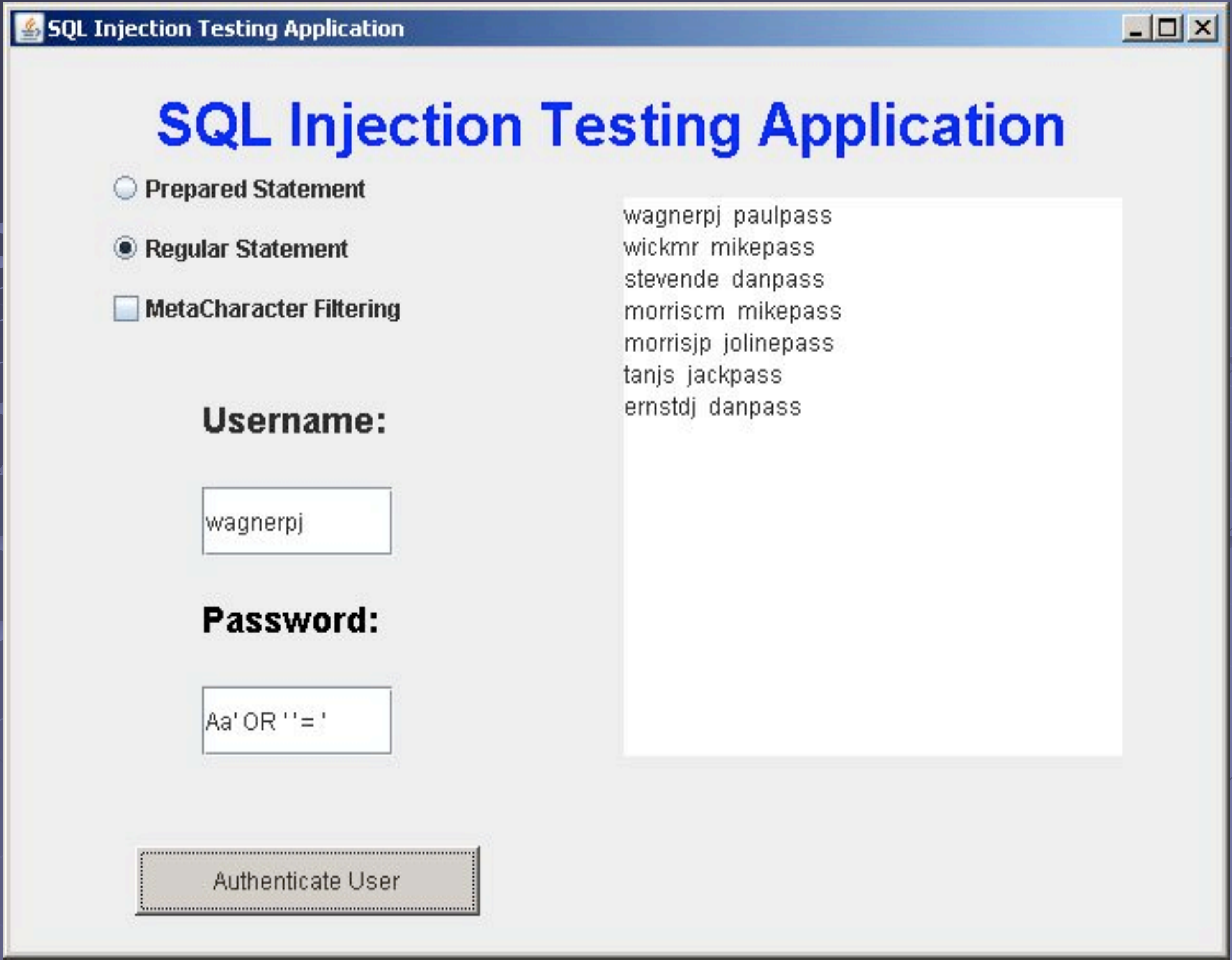
SQL Injection (3)

- Application is expecting one username/password pair row to be returned if success, no rows if failure
- Common variant to simplify the above check:

  -
    ```
    SELECT COUNT(*) FROM users_table
      WHERE username = ‘someuser‘
       AND password = ‘somepass‘;
    ```

SQL Injection (4)

- Attacker enters:
  - any username (valid or invalid)
  - password of: Aa‘ OR ‘ ‘ = ‘
- Query becomes: SELECT * FROM users_table WHERE username = ‘anyname‘ AND password = ‘Aa‘ OR ‘ ‘ = ‘ ‘;
- Note: WHERE clause => ? and F or T => F or T => T
  - AND has higher precedence than OR
- All user/pass rows returned to application
- If application checking for 0 vs. more than 0 rows, attacker is in



SQL Injection Testing Application

SQL Injection Testing Application

( ) Prepared Statement

(●) Regular Statement

☐ MetaCharacter Filtering

Username: wagnerpj

Password: Aa' OR ''='

```
wagnerpj  paulpass
wickmr  mikepass
stevende  danpass
morriscm  mikepass
morrisjp  jolinepass
tanjs  jackpass
ernstdj  danpass
```

SQL Injection - Prevention

- What's the problem here?
  - Not checking and controlling input properly
    - Specifically, not controlling string input
  - Note: there are a variety of ways SQL injection can happen
    - Regular inclusion of SQL metacharacters through
      - Variable interpolation
      - String concatenation with variables and/or constants
      - String format functions like sprintf()
      - String templating with variable replacement
    - Hex or Unicode encoded metacharacters

SQL Injection Prevention (2)

- How to resolve this?
  - First Possible Solution: Check Content
    - Client code checks to ensure certain content rules are met
    - Server code checks content as well (why?)
    - Specifically – don't allow apostrophes to be passed
    - Problem: there are other characters that can cause problems; e.g.

--	// SQL comment character
;	// SQL command separator
%	// SQL LIKE subclause wildcard character

    - Which characters do you filter (blacklist) / keep (whitelist)?
      - Question: is it better to blacklist or whitelist?

SQL Injection – Variant 1

- Any username, password: ‘ or 1=1--
  - Note: -- comments out rest of line, including terminating single quote in application
- Query becomes: SELECT * FROM users_table WHERE username = ‘anyname‘ AND password = ‘ ‘ OR 1=1--‘;
- Note: “OR 1=1” can be appended to any numeric input and force evaluation to true
  - Numeric SQL injection vs. String SQL injection

SQL Injection – Variant 2

- Any username, password: foo’;DELETE FROM users_table WHERE username LIKE ‘%
- Query becomes: SELECT * FROM users_table WHERE username = ‘anyname‘ AND password = ‘foo‘; DELETE FROM users_table WHERE username LIKE ‘%’
- Note: system executes two statements
  - SELECT * FROM users_table WHERE username = ‘anyname’ AND password = ‘foo’; // returns nothing
  - DELETE FROM users_table WHERE username LIKE ‘%’
    - Depending on level of privilege for executing user, we may have trouble here…

SQL Injection – Variant 3

- ODBC (open database connectivity, between any language and any DBMS) allowed shell injection using ‘|’ character
  - ‘|shell(“cmd /c echo “ & char(124) & “format c:”)|’
- Similar issue has existed with MS SQL Server Extended Stored Procedures

SQL Injection – Variant 4

- Second-Order SQL Injection
  - User creates account with user = root'--
    - Application escapes and inserts as root''--
  - User resets password using script/application
    - Query fetches username from database to verify account exists with correct old password; then executes:
    - UPDATE users_table SET PASSWORD='pass' WHERE username = 'root'--'
  - NOTE: above scenario allows user to reset the password on the real root account

SQL Injection – Variant 5

- PL/SQL Cursor Injection
- Structure
  - PL/SQL is procedural language in Oracle built on top of SQL
  - Functions and procedures can be defined which set up cursors that allow execution of dynamically-defined SQL statements
  - Injection techniques can be used with these structures as well
- http://www.databasesecurity.com/dbsec/cursor-injection.pdf

SQL Injection – Variant 6

- Assume web page that gets press releases from db
  - http://www.company.com/pressRelease.jsp?id=5
    - Generates:
  - SELECT title, description, releaseDate, body FROM pressReleases WHERE id = 5
- What if send:
  - http://www.company.com/pressRelease.jsp?id=5 AND 1=1
  - If get press release 5 back, there is a vulnerability
- Now can send:
  - http://www.company.com/pressRelease.jsp?id=5 AND user_name() = 'dbo'
  - If get p.r. 5, know that you're running as user dbo

from HP white paper on Blind SQL Injection

SQL Injection – Interlude

- Many possible variations of SQL Injection
- Many potential metacharacters to filter
  - Dependent on DBMS variety
- Maybe filtering input by blacklisting metacharacters isn't the best approach…
- What is the core problem here?
  - How is SQL injection similar to a buffer overflow?

SQL Injection – Prevention (3)

- Review
  - Regular Statements
    - SQL query (a string) is generated entirely at run-time
    - Custom procedure and data are compiled and run
    - Compilation allows combination of procedure and data, allowing problems with SQL metacharacters

```
String sqlQuery = null;
Statement stmt = null;
sqlQuery = "select * from users where " +
    "username = " + "'" + fe.getUsername() + "'" + " and " +
    "upassword = " + "'" + fe.getPassword() + "'";
stmt = conn.createStatement();
rset = stmt.executeQuery(sqlQuery);
```

SQL Injection – Prevention(4)

- Better Solution
  - Prepared Statements (Parameterized Queries)
    - SQL query is precompiled with placeholders
    - Data is added in at run-time, converted to correct type for the given fields

```
String sqlQuery = null;
PreparedStatement pStmt = null;

sqlQuery = "select * from users where username = ? and
    upassword = ?";
pStmt = conn.prepareStatement(sqlQuery);
pStmt.setString(1, fe.getUsername());
pStmt.setString(2, fe.getPassword());
rset = pStmt.executeQuery();
```

SQL Injection – Prevention (5)

- Issues with PreparedStatements
  - Cannot use them in all situations
    - Generally limited to replacing field values in SELECT, INSERT, UPDATE, DELETE statements
      - E.g. our use for username field value, password field value
    - Example: if also asking user for information that determines choice of table name, cannot use a prepared statement

SQL Injection Prevention (6)

- Additional Precautions
  - Do not access the database as a privileged user
    - Attacker who gains access through user will have that user's privileges
  - Limit database user to only what they need to do
    - e.g. reading information from database, no insert/update/delete
  - Do not allow direct access to database from the internet
    - Require users to go through your (controlled) applications
  - Do not embed database account passwords in your code
    - Encrypt and store them in a repository that is read at application startup
  - Do not expose information in error messages
    - E.g. do not display stack traces

Other Application Issues

- Be aware of how information is transmitted between client applications and database
- Student Research Project at UWEC tested 5 DBMSs and various clients (vendor and user)
  - Most common client applications (vendor-supplied or user-programmed) at least encrypt the connection password
  - Some clients encrypt the connection user
  - Certain DBMSs have varying levels of security (e.g. PostgreSQL)
  - One DBMS transmits the connection password length (MS SQL Server 2005 Express)

Other Application Issues (2)

- Be aware of how application exposes information
- Example: Panel rating system displays each reviewer’s rating for each proposal
  - but doesn’t display individual ratings for a given proposal unless you’ve entered your rating
  - default display is in recommended discussion order, so no information disclosed

Other Application Issues (3)

- However, panel rating system also allows you to sort data by any column, including ratings (average)
- This allows you to see the approximate average rating by other reviewers before entering your rating
- Not a big deal here, but point is that applications may be able to be used in certain ways to expose information

Secure Application Development

- Application Security in the Enterprise Environment
  - J2EE – JDBC, Servlets, JSPs, JNDI, EJBs, …
  - .NET – many components
  - Need to be aware of security issues with each component, interaction of components
- Use of Proxy Applications
  - Assume network filtering most evil traffic
  - Application can control fine-grain behavior, application protocol security

Secure Application Development (2)

- Security Patterns (from J2EE Design Patterns Applied)
  - Single-Access Point Pattern
    - single point of entry into system
    - Example violation: online course system
  - Check Point Pattern
    - centralized enforcement of authentication and authorization
  - Role Pattern
    - disassociation of users and privileges

Secure Operating System

- DBMS interacts with operating system
  - File manipulation
  - Can often run shell commands from DBMS
  - DBMS procedural languages often have packages that interact directly with OS
  - Installation of DBMS uses file system
  - DBMS uses OS resources

Secure Operating System (2)

- Interaction of Oracle and OS
  - Windows
    - Secure administrative accounts
    - Control registry access
    - Need good account policies
    - Others…

Secure Operating System (3)

- Linux/Unix
  - Choose different account names than standard suggestions
  - Restrict use of the account that owns Oracle software
  - Secure temporary directory
  - Some Oracle files are SUID (root)
  - Command line SQL*Plus with user/pass parameters appears under ps output
  - Others…

Secure Web Server

- Interaction of Oracle and Web Server
- Apache now provided within Oracle as its application server, started by default
- Apache issues
  - Standard configuration has some potential problems
    - See Oracle Security Handbook for more discussion
  - Ensure secure communication from web clients to web server
  - Use MaxClients to limit possible connections
  - Others…

Secure Web Server (cont.)

- Internet Information Server (IIS) issues
  - Integration with other MS products (e.g. Exchange Server)
  - Many known vulnerabilities over past versions (patches available)

Secure Network

- Interaction of Oracle and Network
  - Oracle Advanced Security (OAS) product
    - Features for:
      - Authentication
      - Integrity
      - Encryption – use of SSL
  - Oracle server generally behind firewall
    - Good to separate DB and web servers
    - Connections normally initiated on port 1521, but then dynamically selected
  - Other Network Issues To Consider
    - Possibility of hijacking a sys/sysmgr connection
    - Various sniffing and spoofing issues

Miscellaneous Issues

- Newer Oracle Security Features
  - Virtual Private Databases (VPDs)
  - Oracle Label Security
- Auditing
  - Good policy: develop a comprehensive audit system for database activity tracking
    - Database system: can accomplish with triggers
      - On any {read | write }, write to audit log table
    - Can write to OS as well as into database for additional security, accountability for all working with databases

Possible Exercise

- Overall Security Examination of Database System in Networked Environment
  - 1) Database: Set up client(s), test database(s) for:
    - Privileged access accounts
    - Public access through other known/discovered usernames
  - 2) DBMS: Check for known vulnerabilities
    - Check overall system level, patch level
    - Test for specific problems gathered from web search, literature
  - 3) Application:
    - Test for SQL Injection, other application weaknesses

Possible Exercise (2)

- Similar types of tasks for OS, Web Server, Network components
- Task: develop report, including specifics for all areas

Conclusion

- Database system security is more than securing the database; to achieve a secure database system, we need a:
  - Secure database
  - Secure DBMS
  - Secure applications / application development
  - Secure operating system in relation to database system
  - Secure web server in relation to database system
  - Secure network environment in relation to database system

References

- "Oracle Security Handbook" by Theriault and Newman; Osborne/Oracle Press, 2001.
- "Oracle Database Administration: The Essential Reference", Kreines and Laskey; O'Reilly, 1999.
- Pete Finnigan's Oracle Security web site, http://www.petefinnigan.com/orasec.htm
- James Walden's SIGCSE 2006 workshop on "Software Programming Security: Buffer Overflows and Other Common Mistakes"
- Eric Lobner, Matthew Giuliani, Paul Wagner; "Investigating Database Security in a Network Environment", paper published at MICS 2006, http://www.micsymposium.org
- http://www.databasesecurity.com