



# Database Security

*Chapter 8*

*SQL Injection Exploitation and Defense*



# Objectives

- Define SQL injection exploitation
- Identify ways intruders gather information from a network infrastructure
- Describe common strategies for exploiting database infrastructures
- Identify common SQL statements and SQL constructs used to exploit weaknesses
- Apply exploitation for the purpose of identifying infrastructure weaknesses
- Identify defense strategies against SQL injection exploits

# + Exploitation and Information Gathering

## ■ Exploitation

- Act of using system vulnerabilities for the purpose of gaining access or control
- Does not always result in control
  - Depends on effectiveness of SQL query injection techniques and usefulness of generated output
- To defend a system from successful exploitation:
  - Security professional must be aware of means to derive information
    - And what information to protect



# Information That Aids in Exploitation

- Locating a weakness
  - First step in intrusion process
- Example: attacker gains access to database
  - But does not have knowledge of its contents
  - Goal: construct theoretical picture of infrastructure
  - Gathering details aids goal of obtaining access
- Database schema
  - Overall logical structure of objects within the database
  - Includes stored procedures, table, views, and users



# Information That Aids in Exploitation (cont'd.)

- Information about the database
  - Knowing database vendor and version is necessary
- With vendor/version info, attacker can infer:
  - SQL language syntax to use to construct injections
  - Available default procedures
  - Method of processing queries
  - Storage mechanisms utilized
  - Large portion of the schema



# Information That Aids in Exploitation (cont'd.)

- Identifying the vendor
  - Easy for the knowledgeable intruder
  - Multiple clues needed to ascertain database vendor and type
- Clues that aid intruders in identifying vendor
  - The scripting language
  - The platform
  - The database response



# Information That Aids in Exploitation (cont'd.)

- Scripting language
  - Database vendors often lean toward one or two languages
  - Example: PHP used to communicate with MySQL
  - .NET used by SQL Server
  - Oracle has a relationship with Java Script
- Platform
  - Microsoft SQL Server based on foundation of Microsoft Windows Server 2008
    - Indication of Windows points to SQL Server



# Information That Aids in Exploitation (cont'd.)

- Platform (cont'd.)
  - Open source operating systems often used to support MySQL and Oracle
  - Platform provides only one small clue
    - Every configuration is built on unique needs
- Database response
  - Provides most reliable means to identify database
  - Differences in syntax and error format exist between databases
  - Error code can be a valuable piece of information
    - Vendor's Web site provides information on error codes





Vendor	Error code search Web site	Sample
Microsoft SQL Server	<a href="http://www.microsoft.com/technet/support/ee/ee_advanced.aspx">www.microsoft.com/technet/support/ee/ee_advanced.aspx</a>	Login failed for user ' <i>DOMAINNAME\ACCOUNTNAME</i> ' (Error code: 18456)
MySQL	<a href="http://dev.mysql.com/doc/refman/5.1/en/error-messages-server.html">http://dev.mysql.com/doc/refman/5.1/en/error-messages-server.html</a>	error 2003: access denied for user @ localhost (100061)
Oracle	<a href="http://www.ora-code.com/">www.ora-code.com/</a>	ERROR: ORA-01017 invalid username/password; login denied

**Table 8-1 Error code identification**



# Information That Aids in Exploitation (cont'd.)

- Identifying the version
  - Can be equally important to identify as the vendor
  - Gives insight into system capabilities
  - Intruder can take advantage of known vulnerabilities for a given version
- Once database vendor is identified:
  - Locating version number can be an easy task
  - Standard queries used to return version number of that system



# Information That Aids in Exploitation (cont'd.)

- Example of command to discover version in SQL Server:  
`SELECT @@VERSION`
  - Returns version of SQL server, processor, operating system, service pack, and build
- Possible results of injecting statement as a string parameter within a Web application
  - Results returned:
    - If application input or output has not been filtered
    - If expected parameter is a string



# Information That Aids in Exploitation (cont'd.)

- Possible results of injecting statement as a string parameter within a Web application (cont'd.)
  - Error returned:
    - If statement is constructed incorrectly
    - If expected parameter is a number
    - Message may provide the necessary information
  - Nothing returned:
    - If application filters input or output
    - If application is configured to handle error messages in this way
    - Intruder will need to use trial and error approach



# Information That Aids in Exploitation (cont'd.)

- Other types of standard statements
  - Statements to determine database name, location, and language being used
- Administrators should become familiar with these statements
  - Helps understand the amount of information that may be gathered during exploitation

# + Extracting the Real Data

- Techniques presented in previous section allow intruders to gather basic information needed
- Based on knowledge gained about the database:
  - Intruder can construct meaningful queries to gather data
- Exploitation attack equipped to move deeper into the system
  - Targets can be located and data extracted



# Statement Exploits

- Endless number of SQL statements can be injected into the database
- Intruder has capability to access database as a typical user
  - Access is restricted by privileges of the user
- Next section explores common statements used in SQL injection attacks
  - Assumes attacker is working under restricted conditions



# Using UNION

- UNION statements
  - Powerful tools of SQL injection attacks
  - Intruder attaches his or her own queries onto preexisting legitimate statements
- UNION operator combines two or more SQL statements
- Revisiting Yum grocer example

*`http://www.yum.com/index.asp?category=dairy union select Table_Name from Information_Schema.Tables- -`*



# + Using UNION (cont'd.)

- Both results will be combined and returned
  - Provided syntax is correct
- *Table\_Name* call is asking the database to provide names of all the tables for a specific database
- Double dashes at the end comment the rest of the statement
- When using UNION statements
  - Data type and number of columns must be the same as data type and number of columns returned by the original statement

# + Using UNION (cont'd.)

- Method to determine how many columns are being used in original query
- Keep adding null expressions in place of the SELECT statement in the URL until error messages disappear

*http://www.yum.com/index.asp?category=dairy union select null from Information\_Schema.Tables- -*

*http://www.yum.com/index.asp?category=dairy union select null null from Information\_Schema.Tables- -*

*http://www.yum.com/index.asp?category=dairy union select null null null from Information\_Schema.Tables- -*

## + Using UNION (cont'd.)

- Once number of columns being called in original request has been determined
  - Data type can also be determined using trial and error

- Replace each column one at a time with a specific data type

*`http://www.yum.com/index.asp?category=dairy union select Table_Name, null, null from Information_Schema.Tables- -`*

# + Using UNION (cont'd.)

- Replace each column one at a time with a specific data type (cont'd.)

*http://www.yum.com/index.asp?category=dairy* union select null,  
Table\_Name, null from

Information\_Schema.Tables- -

*http://www.yum.com/index.asp?category=dairy* union select null,  
null, Table\_Name from

Information\_Schema.Tables- -

- For those statements that return an error, the incorrect data type exists
  - Can attempt different data type or leave it as null

# + Using UNION (cont'd.)

- Automated tools can assist with this effort
  - Can be found by searching online
  - Tools can be used in cases where *null* cannot be used:
    - Or columns are too large
- *Table\_Name* can be replaced with *Column\_Name* and same strategy applied to identify and view all columns in the database

- Conditional statements
  - If a specified condition is true, a specified action should be taken
  - If a specified condition is false, an alternative action should be taken
  - Advantageous in situations where UNION statements are not allowed
- Example of question intruder could ask the database using a conditional statement:
  - Am I the Administrator?

# + Using Conditions (cont'd.)

- Generating error messages
  - Could alert system administrator to intruder's actions
- Initiated delays
  - Time-based responses
  - Effective means to find an answer to a constructed conditional statement
  - Intruder creates a conditional statement that includes a delay in the response
    - Intruder can infer the answer based on presence or absence of a pause



Platform	Procedure	Comments
Microsoft SQL Server	WAIT FOR DELAY '0:0:5'	Delays a system for five seconds
MySQL	BENCHMARK(100000, encode('Hello '))	Causes the system to encode the word <i>hello</i> one million times, resulting in a delay; the exact time of the delay can be determined by practicing the commands prior to using them
Oracle	Pg sleep(5)	Delays a system for five seconds

Table 8-2 Initiating server time delays



# + Using Conditions (cont'd.)

- Disadvantage of using time delays
  - Delay itself causes process to take longer
- Another method of using conditional statements
  - Designing statements to return different results for true statements as for false statement
  - Might not be effective depending on database defense strategies

# + Large-Scale Extraction

- Strategies presented so far can be tedious and time consuming
  - Effective at providing intruders with data necessary to extend their search
- As an intruder begins to understand a data scheme:
  - Strategies presented can be combined to extract data on a much larger scale

# + Large-Scale Extraction (cont'd.)

- Obtaining database names
  - First step is to list accessible databases
  - Then, one or several can be targeted
  - Tables and columns can be extracted
- Statements that can be used to extract database names
  - Vary depending on the platform
  - Listed in Table 8-3



Platform	Statement	Comments
Microsoft SQL Server	SELECT name FROM sysdatabases	Returns a list of databases; the <i>sysdatabases</i> is installed with SQL Server and contains entries for each major database, which includes the <i>master</i> , <i>model</i> , <i>msdb</i> , and <i>tempdb</i>
MySQL	SELECT schema_Name FROM information_schema.schemata	Returns a list of databases; <i>Information_schema</i> allows access to the metadata of the databases and <i>.schemata</i> is a table that holds information about the databases
Oracle	SELECT global_name FROM global_name	Within Oracle, a user can only maintain one connection to one database, so the only list that can be retrieved is that of the name of the current database

Table 8-3 Database discovery

# + Large-Scale Extraction (cont'd.)

- Obtaining table names
  - Approach: find the table that holds the number and names of all tables in that particular database
- Sysobjects
  - Name of the table holding table information
- Statements that can be used to obtain table names
  - Vary depending on the platform
  - Listed in Table 8-4



Platform	Statement	Comments
Microsoft SQL Server	SELECT name FROM systables	Returns a list of tables found in the <i>sys.tables</i> view of the target database
MySQL	SELECT Column_Name FROM information_schema .tables	Returns a list of databases; <i>Information_schema</i> allows access to the metadata of the databases and <i>.tables</i> is a table that holds information about the tables
Oracle	Select Table_Name from all_tables;	Returns all of the tables in the database

Table 8-4 Table identification

# + Large-Scale Extraction (cont'd.)

- Obtaining columns
  - Intruder can construct statements to extract the columns
    - And the data from within the columns
- Examples of statements that can be used to extract columns
  - Listed in Table 8-5



Platform	Statement	Comments
Microsoft SQL Server	SELECT name FROM syscolumns	Returns a list of tables found in the <i>sys.columns</i> view of the target database
MySQL	SELECT Column_Name FROM information_schema.columns	Returns a list of databases; <i>information_schema</i> allows access to the metadata of the databases and <i>.columns</i> is a table that holds information about the columns
Oracle	Select column_name from all_tab_columns;	Returns all of the columns in the database

Table 8-5 Identifying columns



# + Advanced Techniques

- Filters often used in Web applications:
  - To identify and defer an injection
- Characteristics of filters
  - Search for and block certain known SQL injection characters or statements
- Methods used by intruders to evade and trick filters
  - Encoding
    - Changing characters from ASCII to a different format, such as hex

# + Advanced Techniques (cont'd.)

- Methods used by intruders to evade and trick filters (cont'd.)
  - Case sensitivity
    - Change case of keywords to avoid detection
    - Example: uNiON instead of UNION
  - Breaking it down
    - Break up the common word using URL code
    - Example: 'S' + 'ELE' + 'CT
  - Using alternative statements
    - Example: Case statements instead of *If...Then* statements



# Exploitation of Privileges and Passwords

- Success of techniques previously described:
  - Depends on permissions of the user for which injection statements are being processed
  - Can be limiting to intruders
- Privileges must be identified
  - And increased if possible

# + Identifying Privileges

- Intruder must first know which privileges are grantable on the system
  - Must locate and view privilege tables
- SQL statements used to identify available grantable privileges on the database
  - Listed in Table 8-6



Platform	Statement	Comments
Microsoft SQL Server	EXEC sp_helprotect NULL, 'myusername'	Returns a list of all permissions for the username provided in single quotes, for that particular database. It is necessary to change <i>myusername</i> to the name of the current user; therefore, the current username must be found first for this statement to be effective
MySQL	SELECT Grantee, privilege_type, is_grantable FROM information_schema.user_privileges	Returns a list of grantable privileges on the current database; <i>Information_schema</i> allows access to the metadata of the databases and <i>.user_privileges</i> is a table that holds the grantable privileges
Oracle	Select * from user_sys_privs Select * from user_role_privs Select * from user_tab_privs Select * from user_col_privs	There are four different types of privileges within Oracle: System, Role, Table, and Column. <i>Sys_privs</i> holds all current user-grantable system privileges; <i>Role_privs</i> holds all current user-grantable roles; <i>Tab_privs</i> holds all current user-grantable table privileges; <i>Col_privs</i> holds all current user-grantable column privileges

**Table 8-6 Identifying grantable privileges**

# + Obtaining Passwords

38

- User passwords are stored using nonreversible hash within a table
  - Privileges are needed to access the table
- Password hash
  - Cryptology-encoded string version of a user or system password
  - More difficult to extract
- Some passwords are saved as text strings
  - Pose a great risk to data integrity
  - Can be easily extracted



Platform	Statement	Comments
Microsoft SQL Server	2000- SELECT name.password FROM master.dbo.sysxlogins 2005- SELECT password_hash FROM sql_logins	In SQL Server 2000, login information is stored in the <i>sysxlogins</i> table of the master database; this table was discontinued in later versions and the login information was moved to the <i>sql_logins</i> table
MySQL	SELECT user, password FROM mysql.user	Passwords are located in the <i>mysql.user</i> table in MySQL
Oracle	SELECT name, password FROM sys.user\$ where type#=1	Text passwords can be found in <i>sys users</i> in the <i>system.mgmt_credentials2.table</i> by default

Table 8-7 Extracting passwords

# + Obtaining Privileges

- Obtaining passwords
  - One method of obtaining higher privileges
- Tables that hold password information require high privileges themselves
  - Back-door strategies exist for viewing these tables
- Obtaining administrator privileges
  - Vendor and version specific
  - Strategies vary greatly



# + Obtaining Privileges (cont'd.)

- Brute force attacks
  - Iterative trial and error technique using every possible combination
  - Complexity depends on information user is trying to obtain
  - More successful if much background knowledge is obtained prior to the attack
- Automated tools
  - Can be found online
  - Created for gathering information to assist privilege escalation

# + Obtaining Privileges (cont'd.)

## ■ OPENROWSET

- Common procedure available in SQL Server
  - Can be used for escalating privileges
- Allows user to remotely connect to the database to retrieve information as a different user
- Credentials required to log in remotely must match those on the database
- No time-out for failed login attempts
  - Intruder can use brute force strategies to obtain necessary credentials

# + Obtaining Privileges (cont'd.)

## ■ E-Mail

- Most modern database systems use e-mail to alert administrators
- Database will automatically generate e-mail that includes user's password if lost or forgotten
- Attacker can create malicious code to redirect or copy information sent
- Provides a channel for escalating database privileges and obtaining sensitive information



# Defending Against Exploitation

- Defending database systems
  - Learn infrastructure weaknesses
  - Use careful system monitoring and thoughtful action
  - Combine different strategies to create a strong defense

# + Using Bond Parameters

- Bond parameters
  - Placeholders for binding user input
- Once user input is “bonded,” it is placed into a pre-constructed SQL statement
  - Dynamic SQL is not necessary
  - User data is treated as user data alone
    - Cannot affect the SQL statement
- Very effective defense
  - Can only be used for variables which hold data values

# + Using Bond Parameters

Using MySQL, without bind parameters:

```
$mysqli->query("select first_name, last_name"
    . " from employees"
    . " where subsidiary_id = " . $subsidiary_id);
```

Using a bind parameter:

```
if ($stmt = $mysqli->prepare("select first_name, last_name"
    . " from employees"
    . " where subsidiary_id = ?"))
{
    $stmt->bind_param("i", $subsidiary_id);
    $stmt->execute();
} else {
    /* handle SQL error */}
```



# Sanitizing Data

- Word blocking
  - Blocking certain keywords not allowed as input within a Web application
- Word filtering
  - Balance of blocking known keywords and identifying allowed keywords
  - Limit and mask data fields to only those words or characters that should be used as input

# + Restricting and Segregating Databases

- Assign privileges on the database with greatest granularity available
  - Do not overlook permissions of users on the Web server
- Injections are processed with Web application privileges
  - Web server should have most restricted privileges possible
- Web server and database server should be segregated



# + Security-Conscious Database Design

- During data infrastructure design:
  - Consider where things are stored and how this is viewed by an outsider
  - Consider how objects are named and what information is given away
- Change default names on sensitive data
- Do not use object views for critical database objects
- Create a honeypot environment
  - Fake environment that includes false data to mislead intruders



# Diligent Monitoring

50

- Equally as important as all other security techniques
- Tracking errors can show indications of real-time attacks
- Tracking resource usage can uncover attacker using automated tools
  - Database queries per minute
  - Database requests per minute
- Use baselines to create thresholds for alerts

- Understanding the database schema an important goal of an SQL injection attack
  - More information increases likelihood intruder can obtain control
- Database errors that include vendor's name can provide useful information to intruders
- UNION statements are powerful tools for an attack
  - Intruder can append malicious code onto legitimate SQL statements
- Conditional statements can help intruders gain access

# + Summary (cont'd.)

- Several techniques exist for intruders to evade Web application filters
- Privileges can be escalated by obtaining passwords through brute force attacks, automated tools, e-mail redirection, and OPENROWSET
- Bond parameters are a defense against exploitation
- Segregating database servers ensures that they can be closely monitored and secured