# CS 580 Client-Server Programming
# Spring Semester, 2007
# Doc 15 Security
# April 10, 2007

# References

SQL Injection - http://en.wikipedia.org/wiki/SQL_injection
Buffer Overflow - http://en.wikipedia.org/wiki/Buffer_overflow

NIH Security Web Site http://www.alw.nih.gov/Security/security.html

Applied Cryptography Second Edition, Bruce Schneier, John Wiley & Sons, 1996

Red Team versus the Agents, Scientific American, December 2000, pp. 20, 24.

# Security ≠ Cryptography

Kevin Mitnick often got people’s passwords by asking

# Some Problems Require Global Solution

Denial of Service Attacks

# Some Bad Ideas

Security by Obscurity
Security in the wrong place
Authentication without checking
Back doors

# Security through Obscurity

Security relies on encryption/authentication methods are not obvious

- Reverse the byte order of a message
- Swap bytes in some "secret" way
- Add garbage to data
- Use some "secret" algorithm

Just because you cannot break the encryption does not mean others can't

# Security in the Wrong Place

Regardless of what client does server must authenticate/check

# Back doors

Programmers have the tendency to add debug code to their servers to make testing easier.

This debug code may circumvent any security features of the server.

Example - sendmail "WIZARD"

- Wizard command gave full root privileges to the user
- The default distribution had this command enabled
- The "Internet worm" used this to attack machines throughout the Internet.

Sandia National Labs Security Agents Software

- Agent software based on Lisp
- Agents could perform any Lisp string
- Agents could request other agents to perform tasks
- Intruders could masquerade as an agent

# Some Common Attacks

Buffer Overflow
SQL Injection
Running scripts

# Buffer Overflow

Overflow a buffer to

- change data in other variables
- Execute code from buffer

# Buffer Overflow Example Code

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
  char buffer[10];
  if (argc < 2)
  {
    fprintf(stderr, "USAGE: %s string\n", argv[0]);
    return 1;
  }
  strcpy(buffer, argv[1]);
  return 0;
}
```

Source http://en.wikipedia.org/wiki/Buffer_overflow

# Buffer Overflow Solution 1

Check the Buffer Size

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
  char buffer[10];
  if (argc < 2)
  {
    fprintf(stderr, "USAGE: %s string\n", argv[0]);
    return 1;
  }
  strncpy(buffer, argv[1], sizeof(buffer));
  buffer[sizeof(buffer) - 1] = '\0';  /* explicitly write a string terminator */
  return 0;
}
```

# Buffer Overflow Solution 2

Use a language that checks for array out-of-bounds errors

Java<br>
Smalltalk<br>
Ruby<br>
Python

# SQL Injection

```
"SELECT * FROM users WHERE name = '" + userName + "';"
```

let username be
a’ or ‘t’ = ‘t

```
SELECT * FROM users WHERE name = 'a' or 't'='t';
```

This is always true

let username be
a’; DROP TABLE users; Select * FROM data where name = ‘a

```
SELECT * FROM users WHERE name = 'a' ’;
DROP TABLE users;
Select * FROM data where name = ‘a’;
```

# Preventing SQL Injection In Java

Replace

```
Connection con = (acquire Connection)
Statement stmt = con.createStatement();
ResultSet rset = stmt.executeQuery("SELECT * FROM users WHERE name = '" + userName + "';");
```

with

```
Connection con = (acquire Connection)
PreparedStatement pstmt = con.prepareStatement("SELECT * FROM users WHERE name = ?");
pstmt.setString(1, userName);
ResultSet rset = pstmt.executeQuery();
```

SQl Injection examples from http://en.wikipedia.org/wiki/SQL_injection

# Running Scripts

Some systems allow users to enter a script to be executed

If you need this be very careful on what a script can do

Text