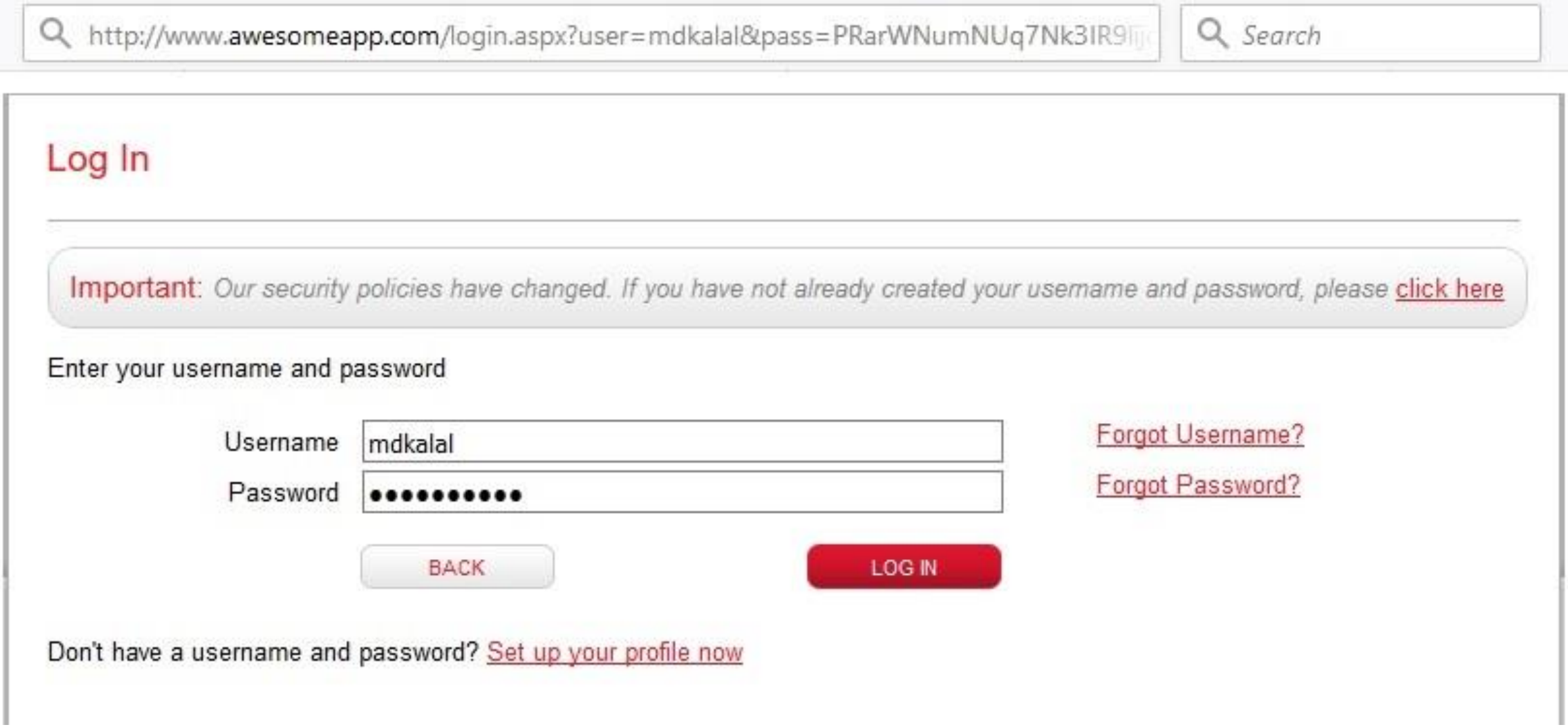
Application Security

Examples


http://www.awesomeapp.com/login.aspx?user=mdkalal&pass=PRarWNumNUq7Nk3IR9
Search
Log In
Important: Our security policies have changed. If you have not already created your username and password, please click here
Enter your username and password
Username
mdkalal
Password
Forgot Username?
Forgot Password?
BACK
LOG IN
Don't have a username and password? Set up your profile now

Server Error in '/' Application.

The network path was not found

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.ComponentModel.Win32Exception: The network path was not found

Source Error:

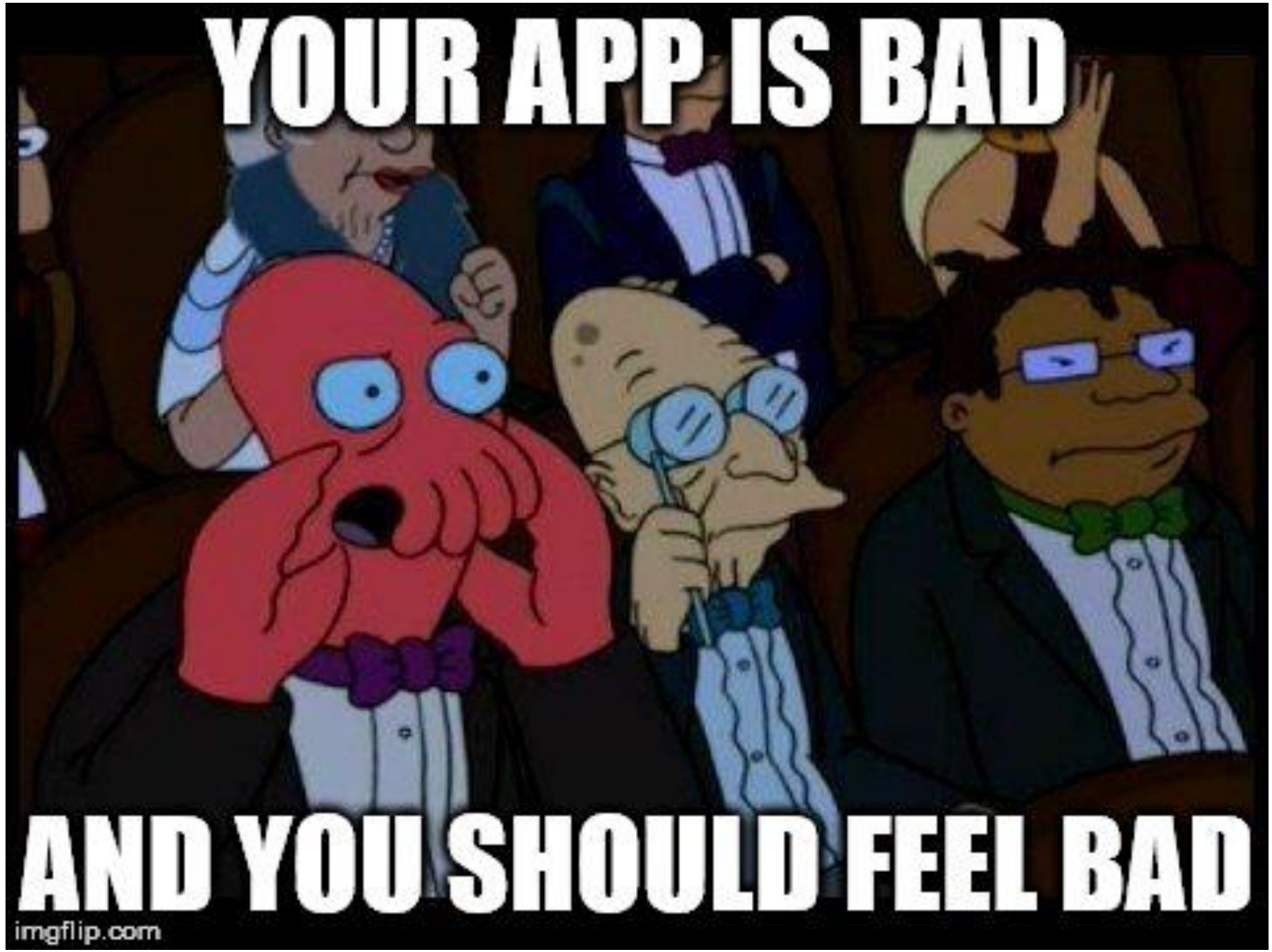
```
Line 63:            string conectionString = "Data Source=192.168.15.200;Initial Catalog=Northwind; Integrated Security=SSPI;";
Line 64:            SqlConnection sqlConnection = new SqlConnection(conectionString);
Line 65:            sqlConnection.Open();
Line 66:            string query = "select uName, uPass from UserDetails where uName = '" + uName + "' and uPassword = '" + uPas
Line 67:
```

Source File: C:\dev\projects\WebApplicationTA\WebApplicationTA\Controllers\AccountController.cs **Line:** 65

Stack Trace:

```
[Win32Exception (0x80004005): The network path was not found]

[SqlException (0x80131904): A network-related or instance-specific error occurred while establishing a connection to SQL Server.
   System.Data.SqlClient.SqlInternalConnectionTds..ctor(DbConnectionPoolIdentity identity, SqlConnectionString connectionOptions,
   System.Data.SqlClient.SqlConnectionFactory.CreateConnection(DbConnectionOptions options, DbConnectionPoolKey poolKey, Object p
   System.Data.ProviderBase.DbConnectionFactory.CreatePooledConnection(DbConnectionPool pool, DbConnection owningObject, DbConnec
   System.Data.ProviderBase.DbConnectionPool.CreateObject(DbConnection owningObject, DbConnectionOptions userOptions, DbConnectio
   System.Data.ProviderBase.DbConnectionPool.UserCreateRequest(DbConnection owningObject, DbConnectionOptions userOptions, DbConn
   System.Data.ProviderBase.DbConnectionPool.TryGetConnection(DbConnection owningObject, UInt32 waitForMultipleObjectsTimeout, Bo
   System.Data.ProviderBase.DbConnectionPool.TryGetConnection(DbConnection owningObject, TaskCompletionSource`1 retry, DbConnecti
   System.Data.ProviderBase.DbConnectionFactory.TryGetConnection(DbConnection owningConnection, TaskCompletionSource`1 retry, DbC
   System.Data.ProviderBase.DbConnectionInternal.TryOpenConnectionInternal(DbConnection outerConnection, DbConnectionFactory conn
```


YOUR APP IS BAD
AND YOU SHOULD FEEL BAD
imgflip.com

Outline

- ❑What is application security?
- ❑Philosophy and practice
- ❑Common vulnerabilities / OWASP examples
- ❑Remediation and prevention strategies

About me

- Mark Kalal
- Software development/technology solutions
- mark.kalal@amtrustgroup.com

Application Security

- Awareness and techniques to prevent the introduction of vulnerabilities in application code
- Making individual apps secure, rather than relying on a firewall or network/server/container “perimeter”
- A way to help developers avoid doing the sort of things that the bad guys exploit

So what?

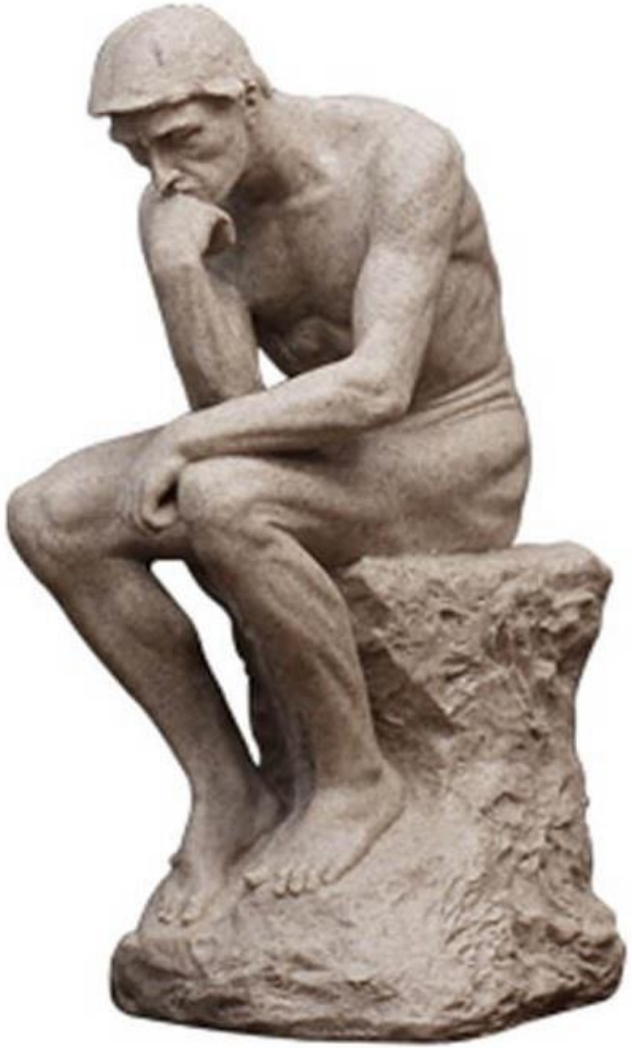
- Insecure code and practice is potentially responsible for any variety of system attacks and data breaches
- Secure coding and testing is challenging
- Important for all developers, but critically so for anyone involved in creating applications that store/process sensitive information


Privacy
Security

codeSecurity === jobSecurity

-Jacob Reynolds

Philosophy



Philosophy

- “Security is an illusion”
- Work is never done, landscape constantly changes
- Consider potential vulnerabilities a *risk*, rather than a *weakness*

Philosophy cont

Tenets include:

- Untrusted data – never assume data/user input is “good”
- Strictest level of permission, fewest users
- Untrusted sources

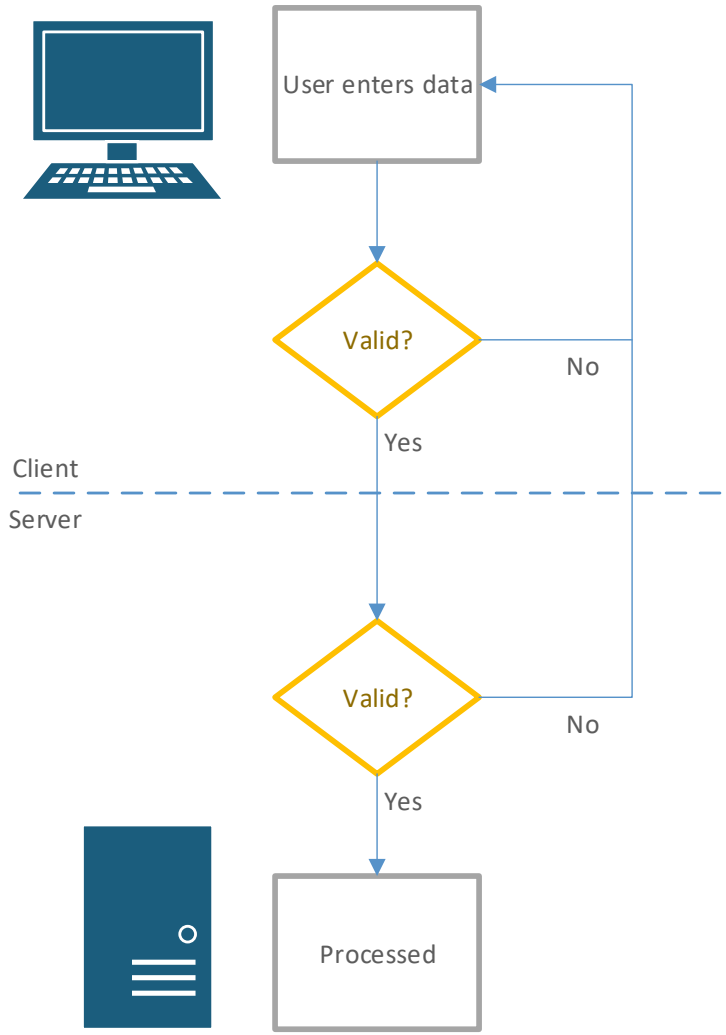
“Security in IT is like locking your house or car – it doesn't stop the bad guys, but if it's good enough they may move on to an easier target.”

— Paul Herbka

Common Vulnerabilities

Unvalidated input

- Validate at the server – client code is more susceptible to manipulation

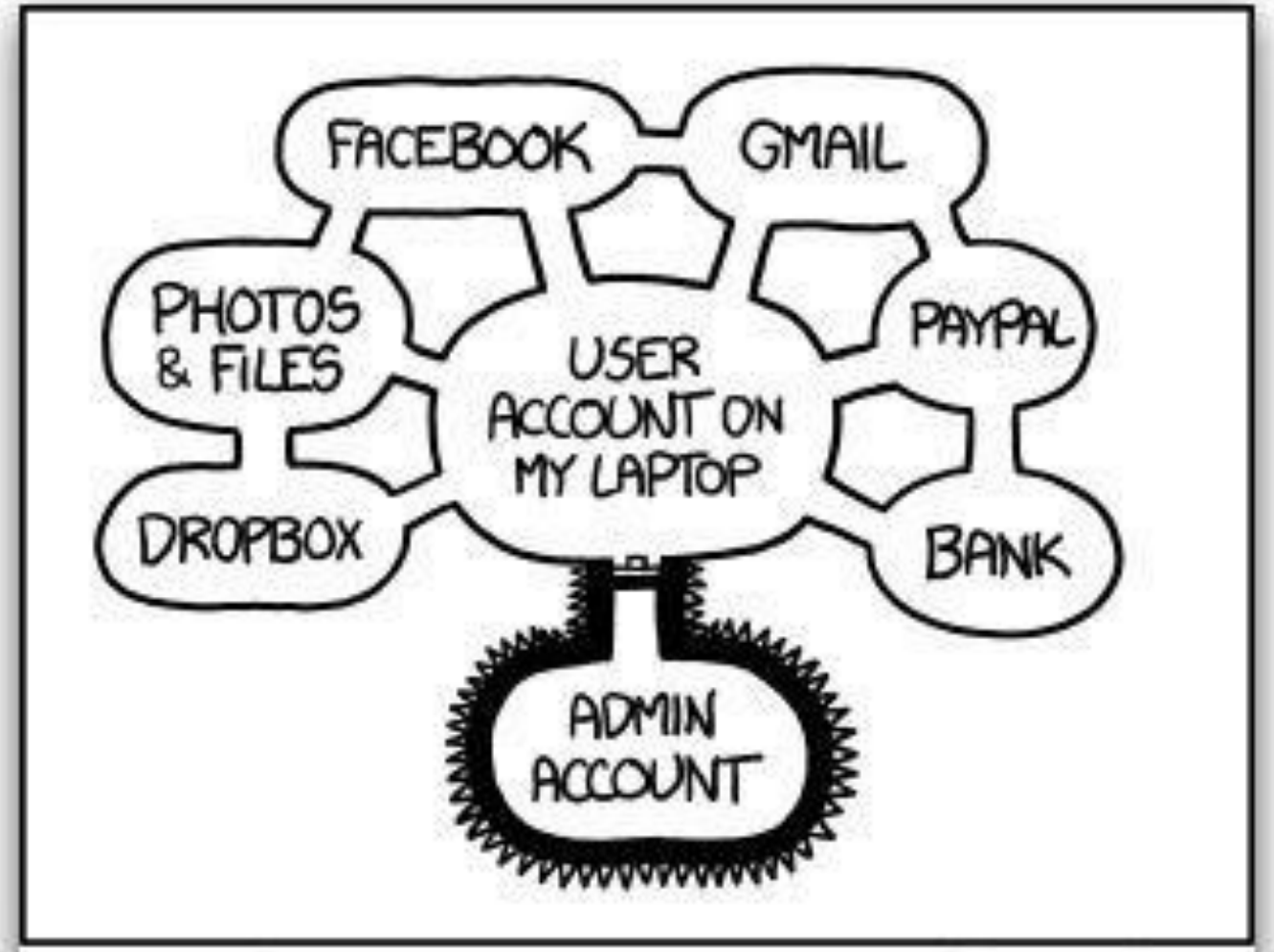


Insecure communications

- HTTPS vs HTTP
- Data unencrypted in transit
- Some websites don’t use SSL/TLS for all content/pages

Weak/no encryption

- Sensitive data can be exposed
- Insecure password storage (salted hash)



IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,

BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.

https://xkcd.com/1200/

OWASP Top Ten Examples

- Injection Flaws
- Broken Authentication
- Cross-Site Scripting
- Using Components with Known Vulnerabilities

Injection Flaws

- Attacker can trick application into executing commands

- Can occur when untrusted data is sent to an interpreter

Injection cont

- Example
  - http://sqlidemo.altervista.org/login1.php

- Prevention – keep untrusted data separate from commands/queries

Parameterized API

```
"INSERT INTO customer_table (last_nm, frst_nm, mid_nm)
     VALUES ( '" varLastNm "' , '" varFrstNm "','" varMidNm '");
```

```
FUNCTION insert_customer
  ( LastNm IN VARCHAR(30),
    FrstNm IN VARCHAR(30),
    MidNm IN VARCHAR(30),
    ErrorCode OUT NUMBER,
    ErrorText OUT VARCHAR(15))
  RETURN  NUMBER
```

Escape characters

Common escapes:

' – single quote

" – double quote

\ – backslash

Query Executed:

```
SELECT * FROM books WHERE title = '' OR author = '\' UNION SELECT * FROM users WHERE \'1\'=\'1';
```

PHP Code:

```
if ($_GET['all'] == 1)
{
    $query = "SELECT * FROM books;";
}
else if ($_GET['title'] || $_GET['author'])
{
    $query = sprintf("SELECT * FROM books WHERE title = '%s' OR author = '%s';",
                            mysqli_real_escape_string($connection, $_GET['title']),
                            mysqli_real_escape_string($connection, $_GET['author']));
}
```

Whitelist

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        if(!request.getParameter("content").matches("[\\w*\\s*]*")){
                request.setAttribute("error", "Please enter only letters, numbers, and spaces.");
                request.getRequestDispatcher("WEB-INF/jsp/render.jsp").forward(request,response);
                return;
        }

        this.content = request.getParameter("content");
        response.sendRedirect(request.getRequestURI());
}
```

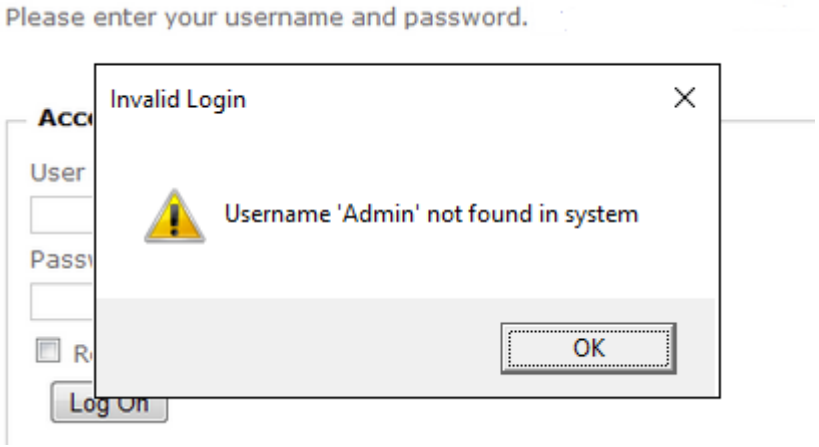
Broken Authentication

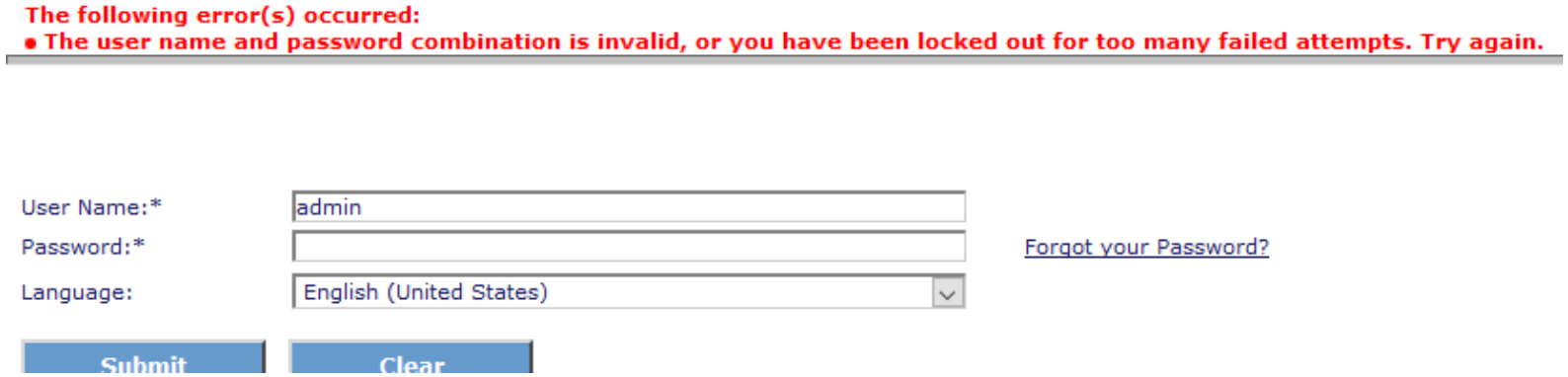
- Authentication, session management is improperly implemented

- Ease of access and exploitation
  - Accounts
  - Sessions

Broken Authentication cont

- Give generic failure messages





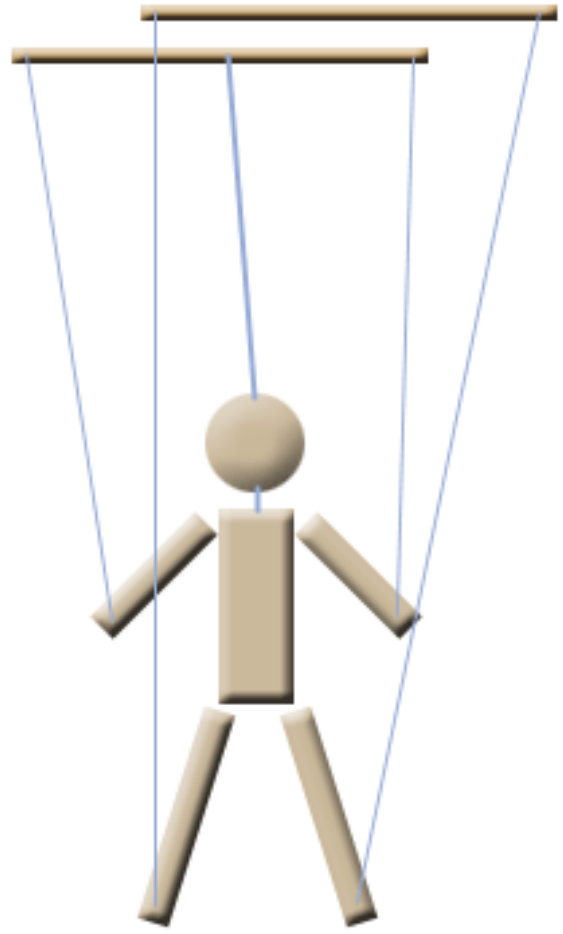
Broken Authentication cont

- Accounts
  - NIST 800-63 B for password policies
  - n3v3r m!nd p@$$w0rd c0mpl3x!ty, passwordLengthAndUniquenessIsMoreImportant
  - Securely store passwords
  - Don’t allow common passwords
- Sessions
  - Enforce strict logout/timeout
  - Renew session IDs often

Cross Site Scripting (XSS)

- Attacker injects code/scripts into web pages, which can be executed on the victims browser

- Can be used to hijack user sessions, deface websites, insert hostile content, redirect users, etc.



XSS cont

- Example
  - https://xss-game.appspot.com/

- Prevention
  - Escape untrusted data / HTML encoding
  - Blacklist/whitelist code “sanitation”
  - Manual code review, automated testing
  - Content Security Policy (CSP)

Using components with known vulnerabilities

- Vulnerabilities and exploits can be found in components / framework libraries.
- Full range of weakness is possible, impacts could be minimal to complete compromise.
- Prevention
  - Keep versions current
  - Monitor updates on public sources
    - https://cve.mitre.org/, https://nvd.nist.gov/
  - Package / dependency manager
  - Establish security focused policies – evaluation prior to implementation, remove unused dependencies, disable unused features, etc.

"To improve security I used a two-factor authentication on my account: my social security number and my credit card number."

Remediation / Prevention

- Education and awareness
- Manual review
- Vulnerability scanning / penetration testing

Remediation / Prevention cont

- Multifactor authentication
- Security as part of each SDLC phase
- Build and maintain your own list of security requirements (ala - https://cwe.mitre.org)

Prevention - Tools

Many available options to help identify potential issues (open source, commercial)

- SonarSource (.net, Java, others)
- FindBugs (Java)

```
269     */
270    public static Account[] getAccounts(String username) throws SQLException{
271        if (username == null || username.trim().length() == 0)
272            return null;
273
274        Connection connection = getConnection();
275        Statement statement = connection.createStatement();
276        ResultSet resultSet =statement.executeQuery("SELECT ACCOUNT_ID, ACCOUNT_NAME, BALANCE FROM ACCOUNTS WHERE USERID = '"+ username +"' ");
277
278        ArrayList<Account> accounts = new ArrayList<Account>(3);
279        while (resultSet.next()){
280            long accountId = resultSet.getLong("ACCOUNT_ID");
281            String name = resultSet.getString("ACCOUNT_NAME");
282            double balance = resultSet.getDouble("BALANCE");
283            Account newAccount = new Account(accountId, name, balance);
284            accounts.add(newAccount);
285        }
286
287        return accounts.toArray(new Account[accounts.size()]);
288    }
289
290    /**
291     * Transfer funds between specified accounts
292     * @param username
```

Problems | Javadoc | Declaration | Console | Servers | Bug Explorer

- AltoroJ 3.1.1 (10)
  - Scary (1)
    - Normal confidence (1)
      - Servlet reflected cross site scripting vulnerability (1)
  - Troubling (9)
    - High confidence (9)
      - Nonconstant string passed to execute or addBatch method on an SQL statement (9)
        - com.ibm.security.appscan.altoromutual.util.DBUtil.addAccount(String, String) passes a nonconstant String to an execute or addBatch method on an SQL statement [Troubling(10), High confider
        - com.ibm.security.appscan.altoromutual.util.DBUtil.addSpecialUser(String, String, String, String) passes a nonconstant String to an execute or addBatch method on an SQL statement [Troubling(1
        - com.ibm.security.appscan.altoromutual.util.DBUtil.addUser(String, String, String, String) passes a nonconstant String to an execute or addBatch method on an SQL statement [Troubling(10), Hig
        - com.ibm.security.appscan.altoromutual.util.DBUtil.changePassword(String, String) passes a nonconstant String to an execute or addBatch method on an SQL statement [Troubling(10), High con
        - com.ibm.security.appscan.altoromutual.util.DBUtil.getAccounts(String) passes a nonconstant String to an execute or addBatch method on an SQL statement [Troubling(10), High confidence]
        - com.ibm.security.appscan.altoromutual.util.DBUtil.getUserInfo(String) passes a nonconstant String to an execute or addBatch method on an SQL statement [Troubling(10), High confidence]
        - com.ibm.security.appscan.altoromutual.util.DBUtil.isValidUser(String, String) passes a nonconstant String to an execute or addBatch method on an SQL statement [Troubling(10), High confidenc
        - com.ibm.security.appscan.altoromutual.util.DBUtil.storeFeedback(String, String, String, String) passes a nonconstant String to an execute or addBatch method on an SQL statement [Troubling(1

Security By Obscurity

- Generally speaking, “not security”
- “With” can be a valid and good strategy (camouflage, decoy)
- Deception technology

Summary

- ✓Defined application security
- ✓Philosophy and practice
- ✓Common vulnerabilities / OWASP examples
- ✓Remediation and prevention strategies

Resources

- OWASP - https://www.owasp.org/index.php/Main_Page
- Top Ten - https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf
- Common vulnerabilities and exposures - https://cve.mitre.org/
- Common weakness enumeration - https://cwe.mitre.org/
- National vulnerability database - https://nvd.nist.gov/
- NIST 800-63B - https://pages.nist.gov/800-63-3/sp800-63b.html
- Injection demo - http://sqlidemo.altervista.org
- XSS game - https://xss-game.appspot.com/
- SonarSource - https://www.sonarsource.com/
- FindBugs - https://sourceforge.net/projects/findbugs/

Thank you!

Feedback welcome

Mark Kalal - mark.kalal@amtrustgroup.com