



build | integrate | secure

Grow with Denim Group



## An Introduction to Application Security

November 11<sup>th</sup>, 2005

# Overview

- Background
- What is Application Security and Why Is It Important?
- Examples
- Where Do We Go From Here?
  - *Assess Critical Infrastructure*
  - *Integrate Security Into the Software Development Lifecycle*
- Questions

# Background

- Denim Group
  - *Texas-based consultancy*
  - *Custom software development*
  - *Systems integration*
  - *Application security*
- Management Team Experience
  - *Large-scale software development*
  - *Air Force information warfare*
  - *Client service for DoD, Big 4, Fortune 500*

# What is Application Security and Why is it Important?

- Application Security Defined
- Fit with General Information Security Landscape
- Why Does Application Security Matter

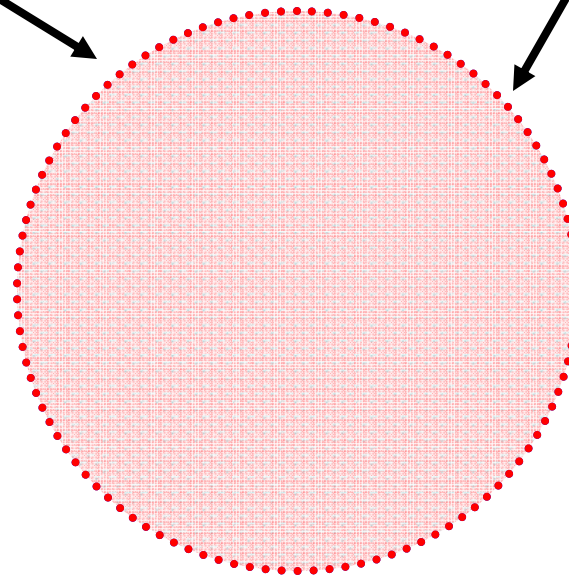
# Application Security Defined

- Ensuring that custom application code performs as expected under the entire range of possible inputs
- Goals:
  - *Confidentiality*
  - *Integrity*
  - *Availability*
- Relationship to Software Quality Assurance
  - *Really a sub-area of SQA*
  - *SQA typically verifies that software does what it is supposed to do*
  - *Application security is concerned that software does not do what it should not do*

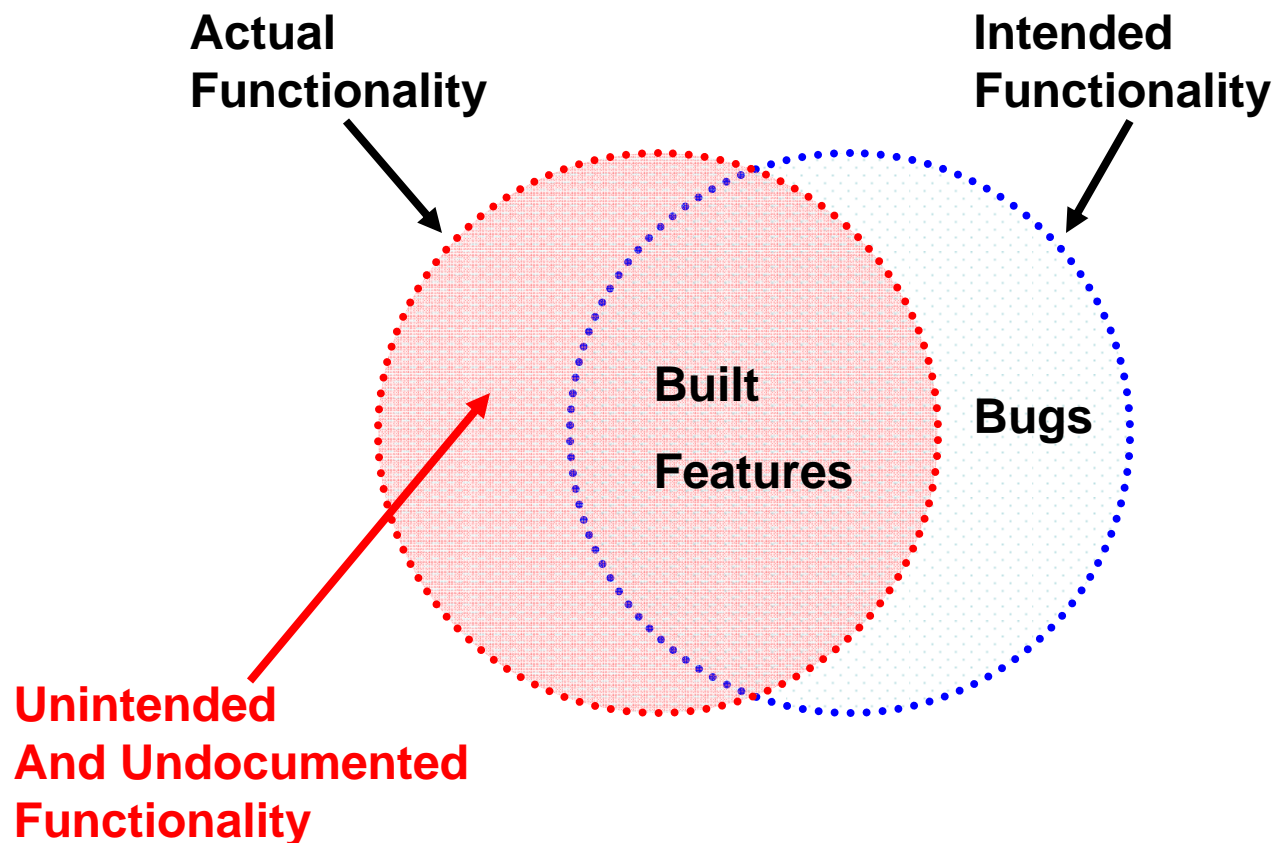
## Software Implementation – Perfect World

**Actual  
Functionality**

**Intended  
Functionality**



## Software Implementation – Real World



# Information Security

- Software Development: Features, functions and timelines
- Information Security: Audit, measure and maintain
- Applies information security principles to custom software development efforts
- Many traditional information security practitioners are ill-equipped to mitigate application security issues

# Why Does This Matter?

- Business-critical web applications are Internet-facing
  - *An increasing number of assets are exposed*
- Most applications have serious flaws
  - *Foundstone and @Stake studies*
- The regulator environment has changed
  - *Sarbanes Oxley*
  - *GLB*
  - *California SB-1386*

# OWASP Top 10 Critical Web Application Security Vulnerabilities

- Unvalidated Input
- Broken Access Control
- Broken Authentication and Authorization
- Cross Site Scripting (XSS)
- Buffer Overflows
- Injection Flaws
- Improper Error Handling
- Insecure Storage
- Denial of Service
- Insecure Configuration Management

<http://www.owasp.org/documentation/topten.html>

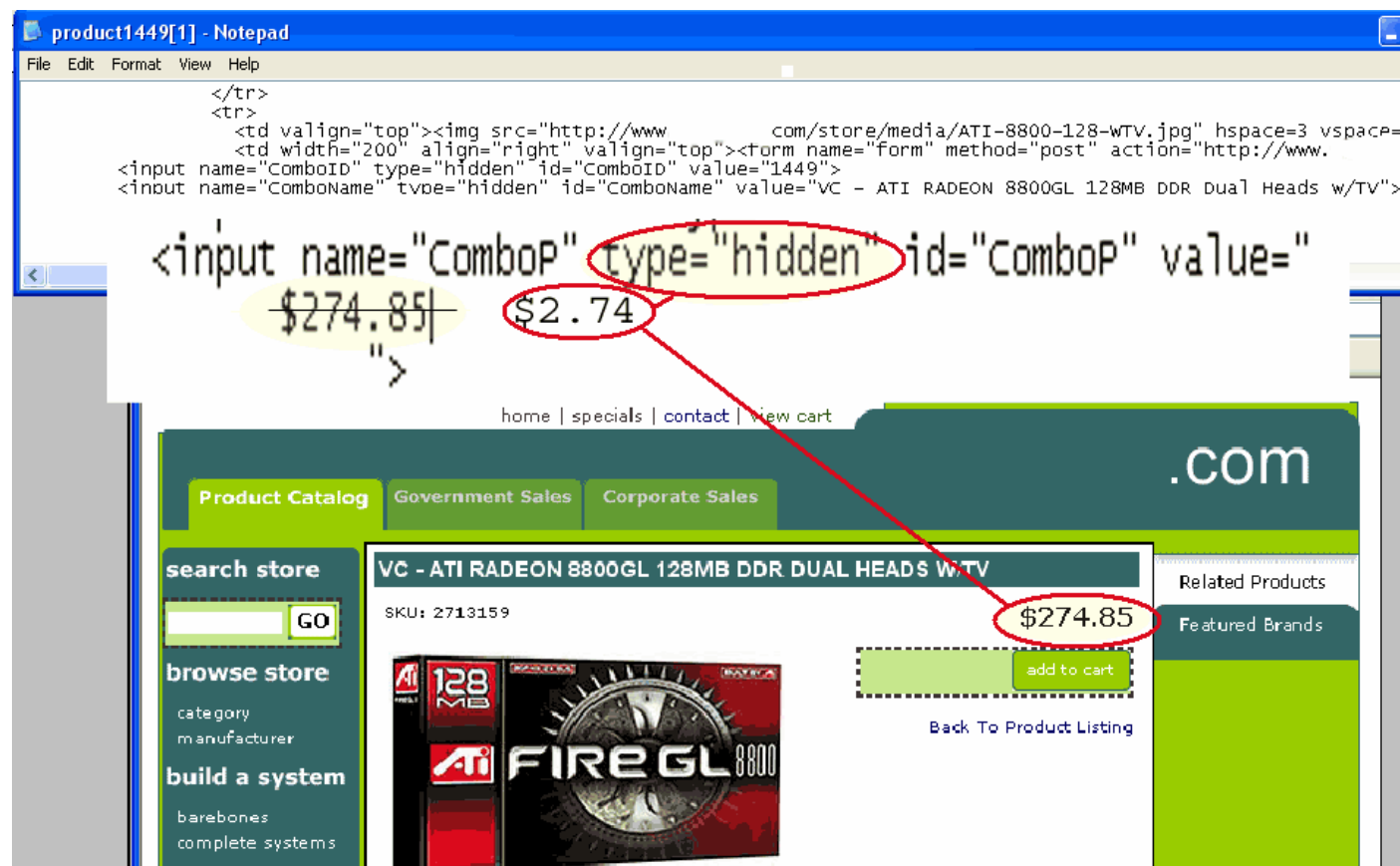
# Examples

- Hidden parameter tampering
- Cookie manipulation
- SQL injection

# Hidden Parameter Tampering

- Price information is stored in hidden HTML form field
- Assumption: hidden field won't be edited
- Attacker edits price parameter
- Attacker submits altered web page with new "price"
- Application trusts the price parameter from the user
- Still widespread in many web stores

# The Attack

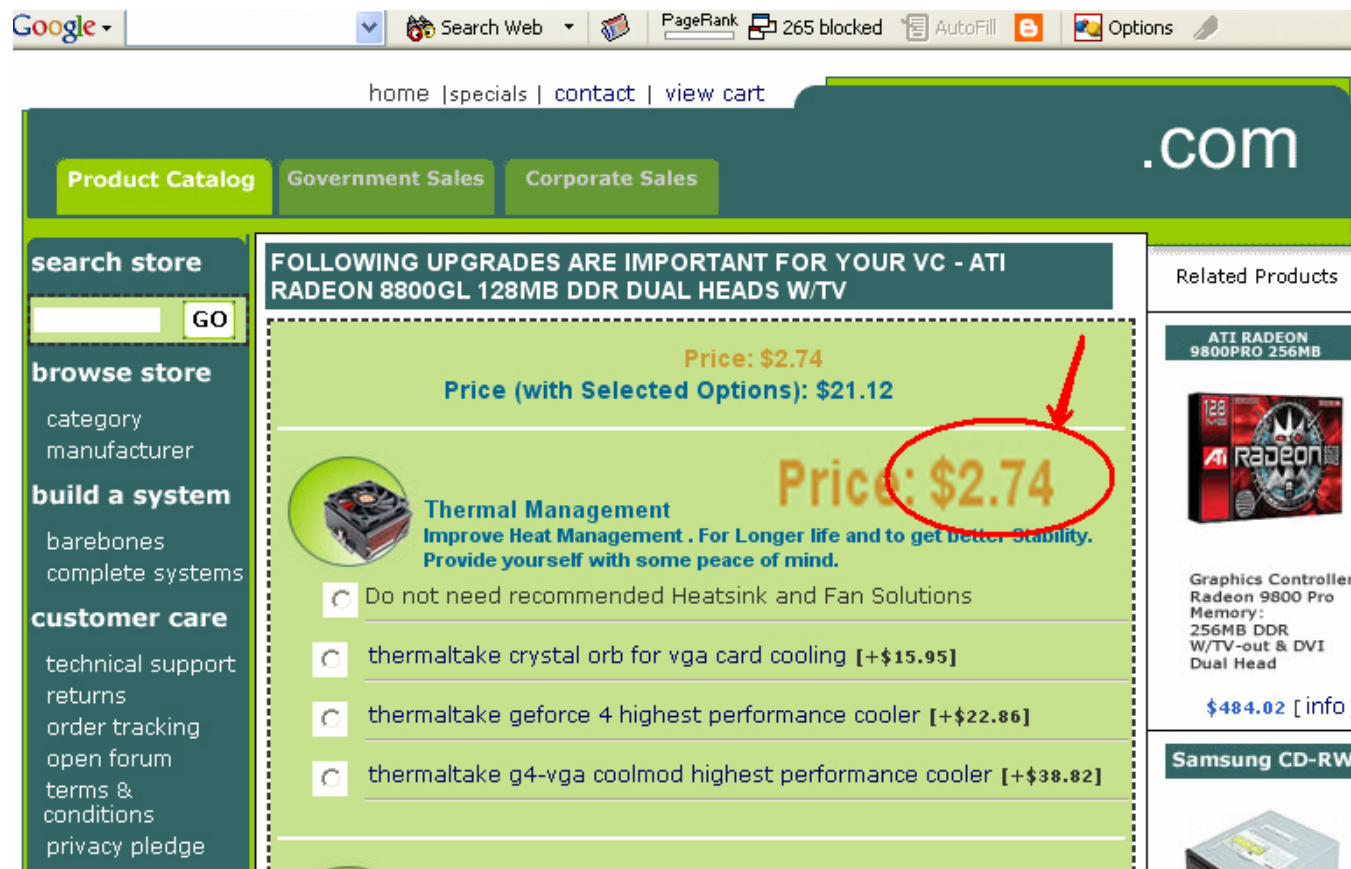


The image illustrates a price change attack. The top part shows a Notepad window with the following HTML code:

```
</tr>
<tr>
  <td valign="top"><form name="form" method="post" action="http://www.
<input name="ComboID" type="hidden" id="ComboID" value="1449">
<input name="ComboName" type="hidden" id="ComboName" value="VC - ATI RADEON 8800GL 128MB DDR Dual Heads w/TV">
<input name="ComboP" type="hidden" id="ComboP" value="
  <del>$274.85</del>
  $2.74
">
```

The code shows a hidden form field named "ComboP" with a value of "\$2.74", which is circled in red. A red arrow points from this field to the product page below. The product page shows the product "VC - ATI RADEON 8800GL 128MB DDR DUAL HEADS WTV" with a price of "\$274.85" (circled in red) and an "add to cart" button. The original price of "\$274.85" is crossed out, and the new price of "\$2.74" is shown below it.

# The Result



home | specials | contact | view cart

Product Catalog Government Sales Corporate Sales .com

search store  
 GO

browse store  
 category  
 manufacturer

build a system  
 barebones  
 complete systems

customer care  
 technical support  
 returns  
 order tracking  
 open forum  
 terms & conditions  
 privacy pledge

**FOLLOWING UPGRADES ARE IMPORTANT FOR YOUR VC - ATI RADEON 8800GL 128MB DDR DUAL HEADS W/TV**

Price: \$2.74  
 Price (with Selected Options): \$21.12


**Price: \$2.74**

**Thermal Management**  
 Improve Heat Management . For Longer life and to get better Stability.  
 Provide yourself with some peace of mind.

- Do not need recommended Heatsink and Fan Solutions
- thermaltake crystal orb for vga card cooling [+\$15.95]
- thermaltake geforce 4 highest performance cooler [+\$22.86]
- thermaltake g4-vga coolmod highest performance cooler [+\$38.82]

Related Products


ATI RADEON 9800PRO 256MB



Graphics Controller: Radeon 9800 Pro  
 Memory: 256MB DDR  
 W/TV-out & DVI Dual Head

\$484.02 [ info ]

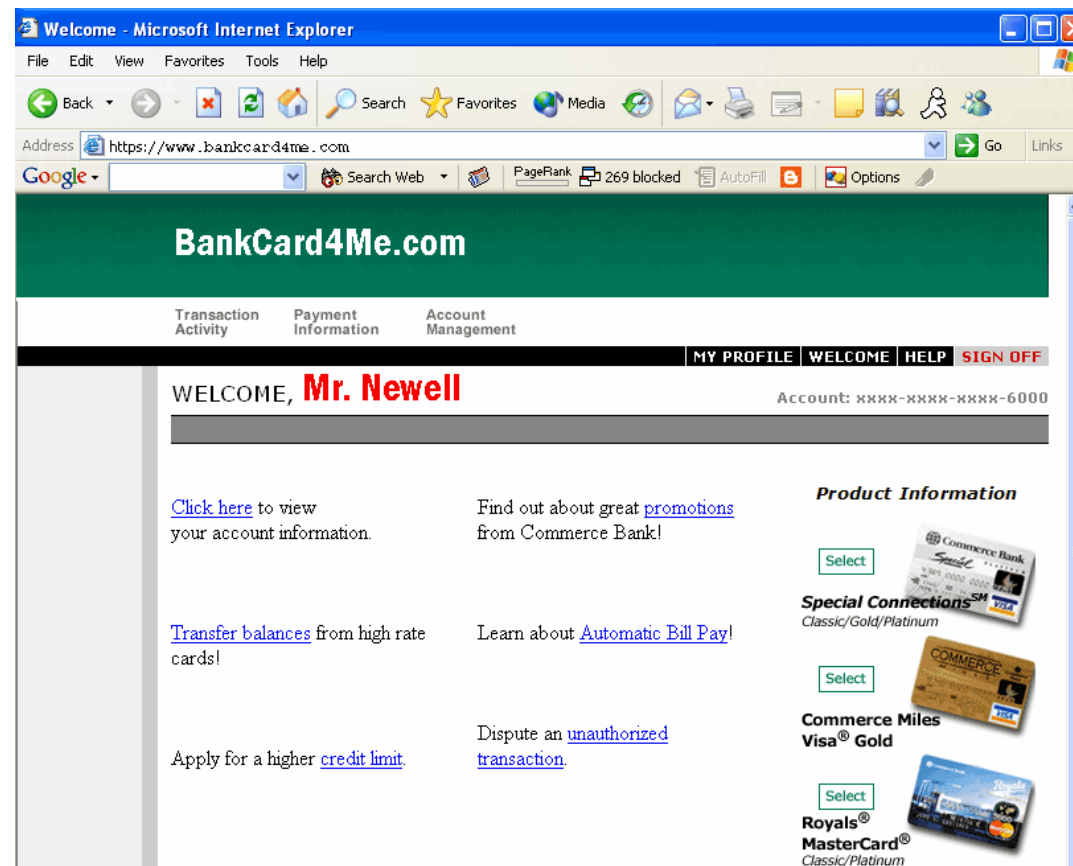
Samsung CD-RW



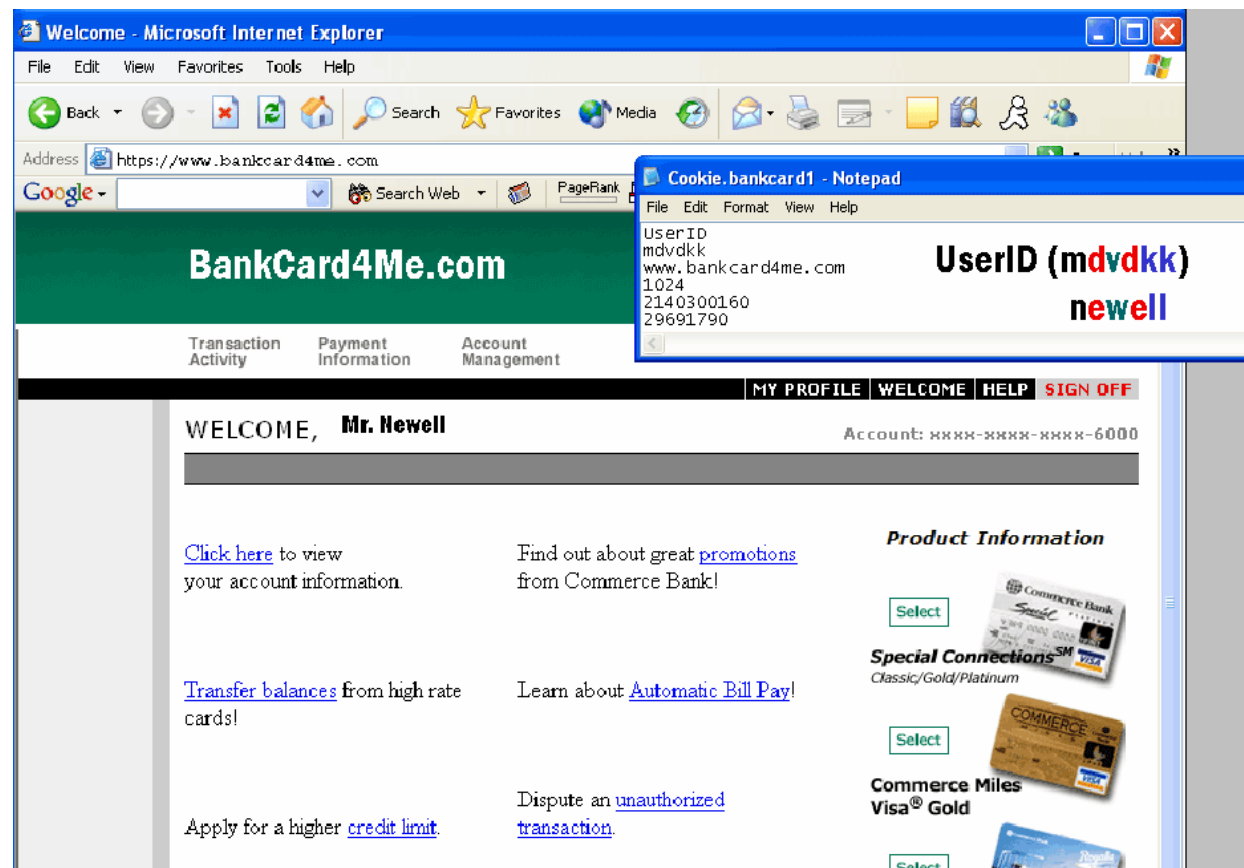
# Cookie Manipulation

- Browser cookie is used to store user identity information
- Assumption: cookies are set by server side code, handled by the browser automatically and not manipulated by users
- Attacker alters cookie
- Application trusts the browser cookie and allows attacker to assume identity of another user

# The Attack



# The Attack



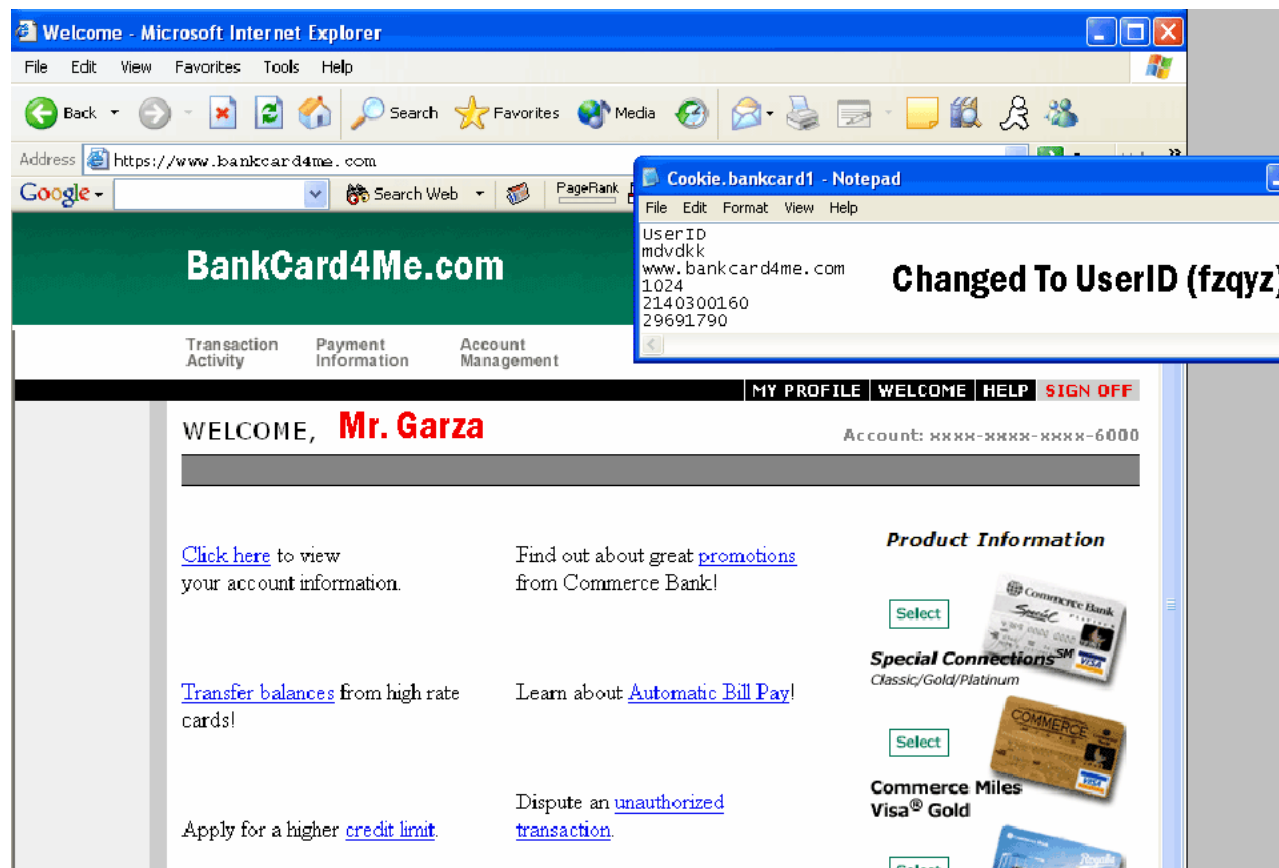
The screenshot shows a Microsoft Internet Explorer browser window displaying the BankCard4Me.com website. The address bar shows the URL <https://www.bankcard4me.com>. The website content includes a navigation menu with "Transaction Activity", "Payment Information", and "Account Management". A welcome message reads "WELCOME, Mr. Newell" with an account number "Account: xxxx-xxxx-xxxx-6000". There are several links for account management and product information.

In the foreground, a Notepad window titled "Cookie.bankcard1 - Notepad" displays the following cookie data:

```
UserID  
mdvdkk  
www.bankcard4me.com  
1024  
2140300160  
29691790
```

To the right of the Notepad window, the text "UserID (mdvdkk) newell" is displayed, indicating the user's identity and the stolen session.

# The Result



Cookie.bankcard1 - Notepad

```
File Edit Format View Help
UserID
mdvdkk
www.bankcard4me.com
1024
2140300160
29691790
```

**Changed To UserID (fzqyz)**

BankCard4Me.com

Transaction Activity    Payment Information    Account Management

MY PROFILE    WELCOME    HELP    SIGN OFF


WELCOME, **Mr. Garza**    Account: xxxx-xxxx-xxxx-6000

[Click here](#) to view your account information.    Find out about great [promotions](#) from Commerce Bank!


[Transfer balances](#) from high rate cards!    Learn about [Automatic Bill Pay!](#)

Apply for a higher [credit limit](#).    Dispute an [unauthorized transaction](#).


**Product Information**

Select 

**Special Connections<sup>SM</sup>**  
Classic/Gold/Platinum

Select 

**Commerce Miles Visa<sup>®</sup> Gold**

Select 

# SQL Injection

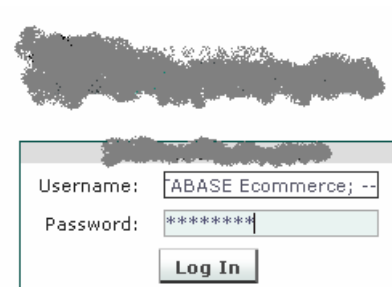
- SQL statements are created from a combination of static text and user inputs
- Assumption: users will enter well-formed inputs
- Attacker crafts a custom input to hijack control of the SQL interpreter and execute arbitrary code
- Very common flaw with tremendous security implications

# The Attack

```
try {
    string username =
    request.getParameter("username");
    string password =
    request.getParameter("password");
    string sSql = "SELECT * FROM User WHERE
    username = " + username + " AND password = " +
    password + """;
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery(sSql);
    ...
} catch(Exception ex) {}
```

# The Attack

- Specially crafted input contains SQL control characters



# The Attack

- Malicious user sends in a username parameter of: Dcornell'; DROP DATABASE Ecommerce; --

## SQL Executed is:

```
SELECT * FROM User WHERE username = 'Dcornell';  
DROP DATABASE Ecommerce; -- AND password =  
'whocares'
```

- Attacker can execute arbitrary database queries with the same permissions as the application
  - *View sensitive data*
  - *Modify data*
  - *Destroy data*

# Where Do We Go From Here

- Assess existing critical infrastructure applications
- Integrate security into the software development lifecycle

# Assess Critical Infrastructure

- Automated scanning tools can help find certain coding flaws but are very poor at finding potentially more damaging design and architecture flaws
- Penetration tests to better evaluate the security state of applications
- Source code analysis to find design and architecture flaws

# Integrate Security Into the Lifecycle

- Train business analysts, application architects, software developers and quality assurance personnel on application security issues
- Integrate security into application requirements
- Threat modeling
- Architecture and design peer review
- Periodic penetration testing
- Also applies to *acquiring* software packages

# Questions

Dan Cornell

[dan@denimgroup.com](mailto:dan@denimgroup.com)

Denim Group, Ltd.

[www.denimgroup.com](http://www.denimgroup.com)