

Web Attacks and Defenses

Vulnerable .Net PetStore Demo

Tyler Shields & Anthony Barkley

July 12, 2004



Where Security & Business IntersectSM

Agenda

- **Introduction**
- **Application Attacks Overview**
- **Attack Methodologies**
- **Demonstration Platform Details**
- **Demo of Common Vulnerabilities**

Cross-site Scripting

SQL Injection

Price Fixing (aka Parameter Tampering)

Directory Traversal

Verbose Error Messages

Information Disclosure

- **Review**
- **Questions**

Introduction

- **@Stake, Inc.**

- Premier digital security consulting company
- Services cover key aspects of security, including applications security, critical infrastructure, wireless and wired networks, storage systems, education, and incident readiness.
- @stake clients include six of the world's top ten financial institutions, four of the world's top ten independent software companies and seven of the world's top ten telecommunications carriers.
- Offices in RTP, Chicago, London, Boston, San Francisco, Seattle, New York City, Denver.



Where Security & Business IntersectSM

Introduction

- **Anthony Barkley**

- Technical Director for @stake's RTP office. He has been working in the security industry for over a decade with his primary focus on security for enterprise and Application Service Provider (ASP) infrastructures. His professional experience includes systems integration, internetworking, security architecture, teaching, and project management from concept to implementation. Anthony holds a BA from North Carolina State University and has completed numerous certifications, including the CCSA, CCSE, MCSA, MCSE and CISSP.

- **Tyler Shields**

- Senior Security Architect with @stake, Inc. Tyler's primary focus is application penetration testing and analysis. He has presented at major industry conferences, taught security courses, and has been involved in the security community for nearly a decade contributing to many prominent mailing lists and open source security projects. Tyler graduated from the Rochester Institute of Technology, in Rochester, NY with a BS in Information Technology and since has acquired a number of security certifications including his CCSA, CCSE, and CISSP.

Application Attacks Overview

- **Application security measures are often implemented at the network level**
 - This will not work effectively as network ACLs cannot protect against the use of valid connections
 - Network security cannot generally provide adequate protection at the application level
 - Network implemented application controls may introduce significant impact on performance

- **Applications are the next generation attack point**
 - Information about application attacks is more readily available
 - Protections against application attacks are not as robust or mature as network security
 - Tools for testing applications are becoming more readily available

Application Attacks Overview

- **Applications are targeted because they are the access point for data, products, resources, and money**
 - Even with conventional network defense, many applications can be attacked
 - Security controls around applications can be bypassed, which allows attackers to:
 - Modify information
 - Steal products
 - Credit money to accounts
 - In some cases the attack may grant access to other portions of the application or to the environment
 - The attack may not even involve theft
 - If the attacker understands the system being attacked, it is possible that they will be able to merely “borrow” resources

Attack Methodology

- **Discover**
- **Target**
- **Attack**

Attack Methodology: Discover

- **Examine the environment**
 - Identify what types and version of applications are running (banners/headers)
 - Identify what ports are open for communication to the server and the application
 - Examine extensions: foo.jhtml, foo.shtml - will often reveal the application server engine (weblogic, coldfusion, tomcat, etc.)
- **Generate and examine errors**
 - Submit ridiculous input and monitor response (fuzzing)
 - Database errors are extremely helpful
- **Look for information left behind from development**
 - Sample code or snippets

Attack Methodology: Discover

- **Look for configuration errors:**
 - Use a sniffer to examine traffic
 - Review client software
 - Use network scanning and probing tools

- **Look for environment errors:**
 - Identify ways to circumvent application security controls
 - Reveal program data flow and logic flow
 - Reveal critical systems that are easier to attack than the original target

Attack Methodology: Discover

- **Use a debug proxy to intercept client-server communications**
 - Observe application traffic and modify components of application traffic independently of the client
 - Look for hidden fields and notes embedded in the source
 - Developers tend to make assumptions about the integrity of “client” generated data, like headers and other data that is supposedly concealed from the user -- abuse these assumptions
- **Include a regular expression engine to make on the fly replacements**
- **Use an existing tool or a write a new one**
 - HTTP: HTTPush, RFProxy, @stake Webproxy

Attack Methodology: Target

- **Login Mechanism**
- **Session Management**
- **Input Fields**
- **Unprotected Interfaces**
- **Application Related Infrastructure**
- **Application Related Networks**

Demonstration Platform Details

- **Microsoft .Net PetShop 3.0 is Microsoft's implementation of a J2EE sample application.**
 - Original application can be found at :
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/psimp.asp?frame=true>
- **Does not out of the box have these vulnerabilities**
 - Added by @stake for demonstration purposes
- **Vulnerabilities based on common web application issues found in production environments.**
- **Platform built on Windows 2000 Server, IIS 5.0 and SQL Server 2000**

Demonstration Platform Details

- **What it is and what does it do**
- **Vulnerability identification**
- **Execution of the vulnerability**
- **Review of faulty application code**
- **Possible mitigation techniques**

Common Vulnerabilities - Cross Site Scripting (XSS)

- **What it is and what does it do**

- XSS is the injection of code or script into a web page that is then displayed to a third party executing the script in the context of their web browser.

- **Vulnerability identification**

- Look for anywhere where client supplied data is redisplayed to the end user and not properly input validated or output encoded.

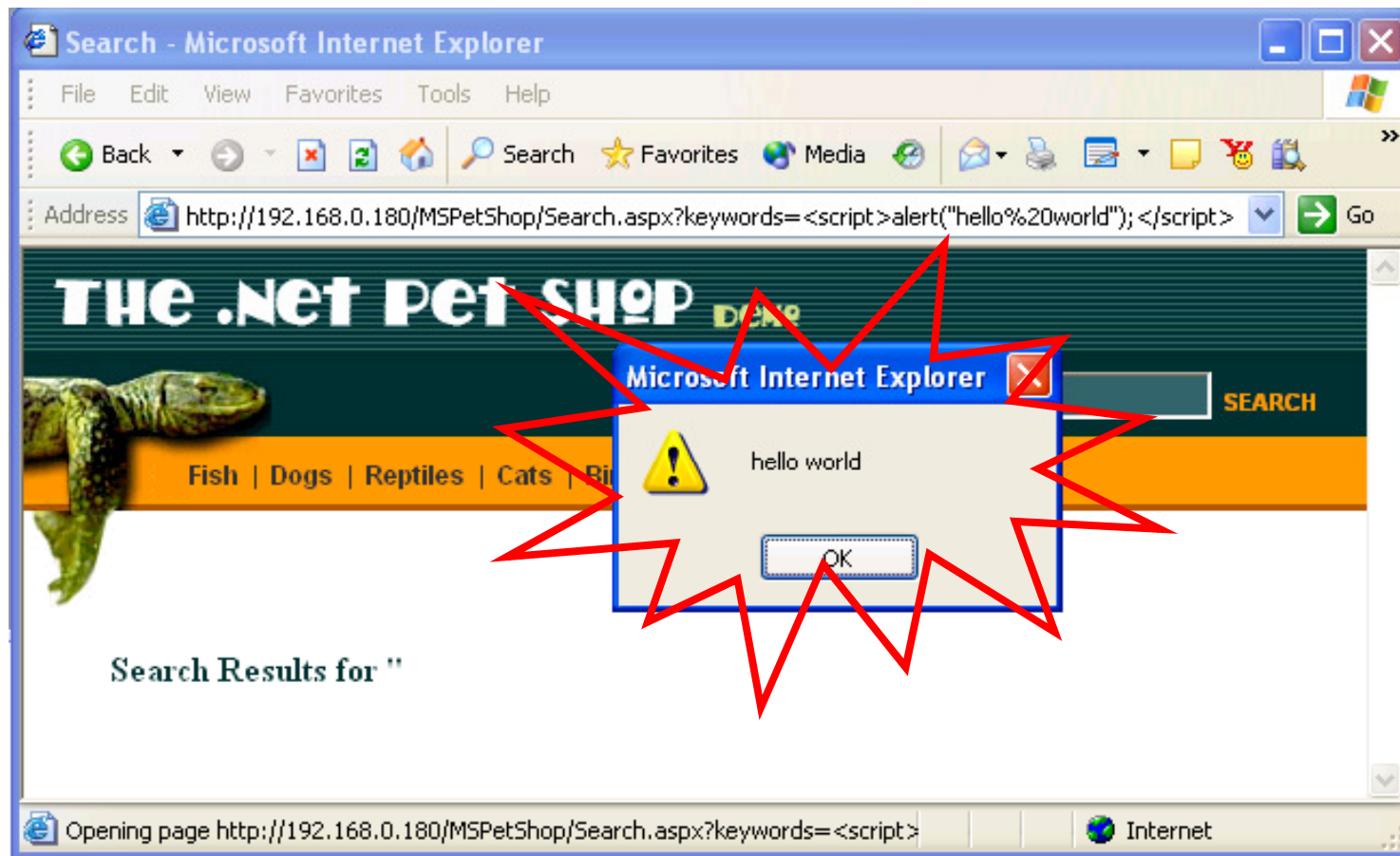
- **Execution of the vulnerability**

- Inject some simple script and see if the resulting page contains your data.

Common Vulnerabilities - Cross Site Scripting (XSS)

- **Review of faulty application code**
 - Search Results for "`<%=Request["keywords"]%>`"
 - Lack of input validation and output encoding
- **Possible mitigation techniques**
 - Input validation and output encoding
 - The original code `htmlEncodes` suspicious values on output
 - Validate all input against a known set of approved characters
 - Defense in depth:
 - Truncate all input fields to a max reasonable length
 - Out-of-the box .Net bans '<'

Cross Site Scripting Demonstration



[http://TARGETIP/MSPetShop/Search.aspx?keywords=<script>alert\('hello%20world'\);</script>](http://TARGETIP/MSPetShop/Search.aspx?keywords=<script>alert('hello%20world');</script>)

Common Vulnerabilities – SQL Injection

- **What it is and what does it do**

- SQL Injection is a vulnerability allowing an attacker to submit SQL statements of their choosing that are then processed by the application.

- **Vulnerability identification**

- Typical identification techniques include submitting characters such as ‘, “, or – characters into an input field. Based on the resulting errors and/or data, SQL Injection points may be identified.

- **Execution of the vulnerability**

- Longer cycle of execution. Based on resulting error pages and/or data the entire database schema and stored data may be compromised.

Common Vulnerabilities – SQL Injection

■ Review of faulty application code

- `string sql = "SELECT Item.ItemId, Item.Attr1, Inventory.Qty, Item.ListPrice, Product.Name, Product.Descn FROM Item INNER JOIN Inventory ON Item.ItemId = Inventory.ItemId INNER JOIN Product ON Item.ProductId = Product.ProductId WHERE Item.ItemId = '"+itemId+"'";`
- Lack of input validation
- Dynamically concatenated SQL statements

Common Vulnerabilities – SQL Injection

■ Possible mitigation techniques

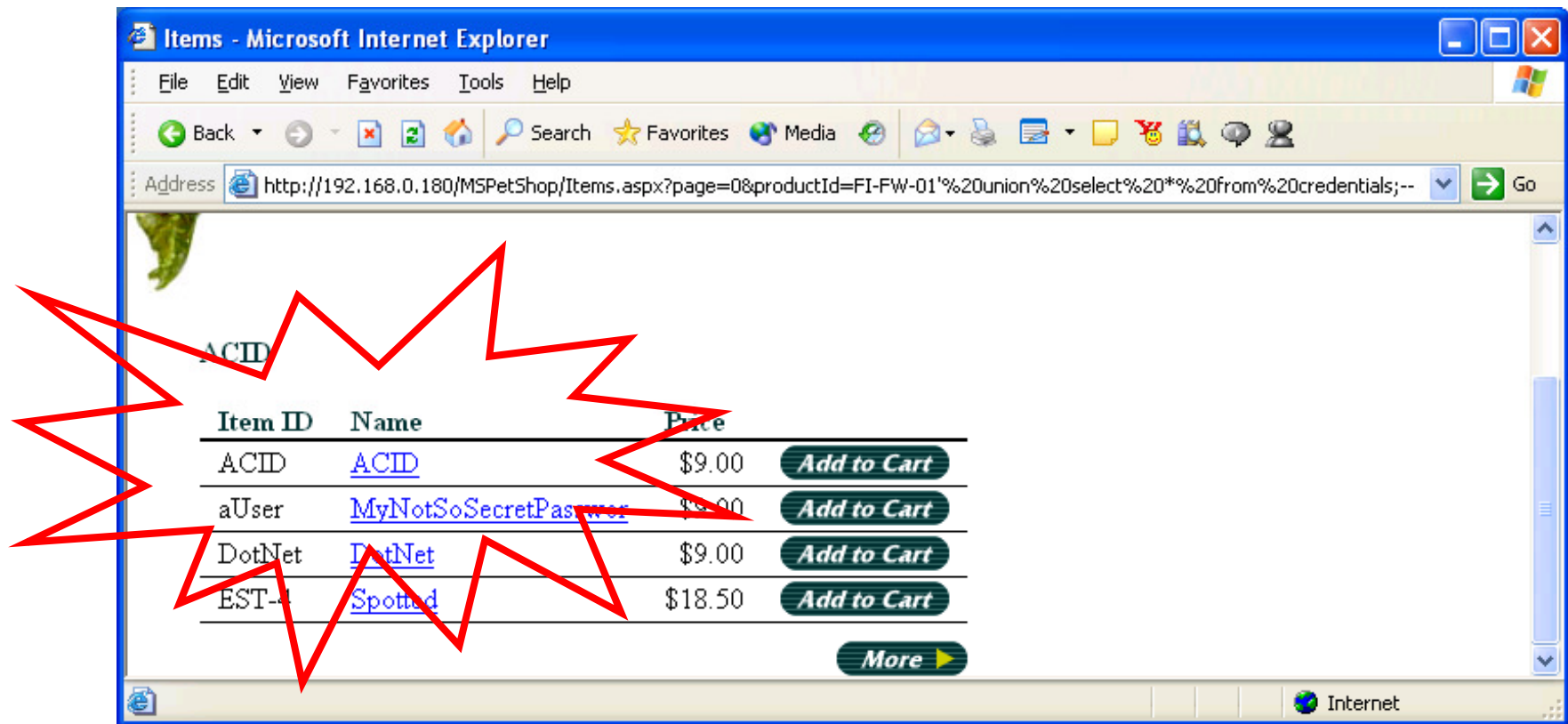
– Input Validation

- The original code htmlEncodes suspicious values such as '
- Input validation: itemId should not contain anything but A-Z,0-9 and not be longer than 15 characters. Use white-list based input filtering.

– Prepared statements

- Arguments are added to the statement after it is parsed for validation
- Limited privileges: Create a “browse” identity with only select privileges on the products tables

SQL Injection Demonstration



http://TARGETIP/MSPetShop/Items.aspx?productId=FI-FW-01'%20union%20select%20*%20from%20credentials;--

Common Vulnerabilities – Price Fixing

- **What it is and what does it do**
 - Commonly referred to as parameter manipulation attacks
 - Allows the attacker to modify values that are then utilized by the application in an unchecked fashion

- **Vulnerability identification**
 - Modification of parameters in a logical fashion in an attempt to verify the validation and authorization routines of the application server

- **Execution of the vulnerability**
 - If possible modify in the URL directly
 - Alternatively utilize an http(s) intercepting proxy

Common Vulnerabilities – Price Fixing

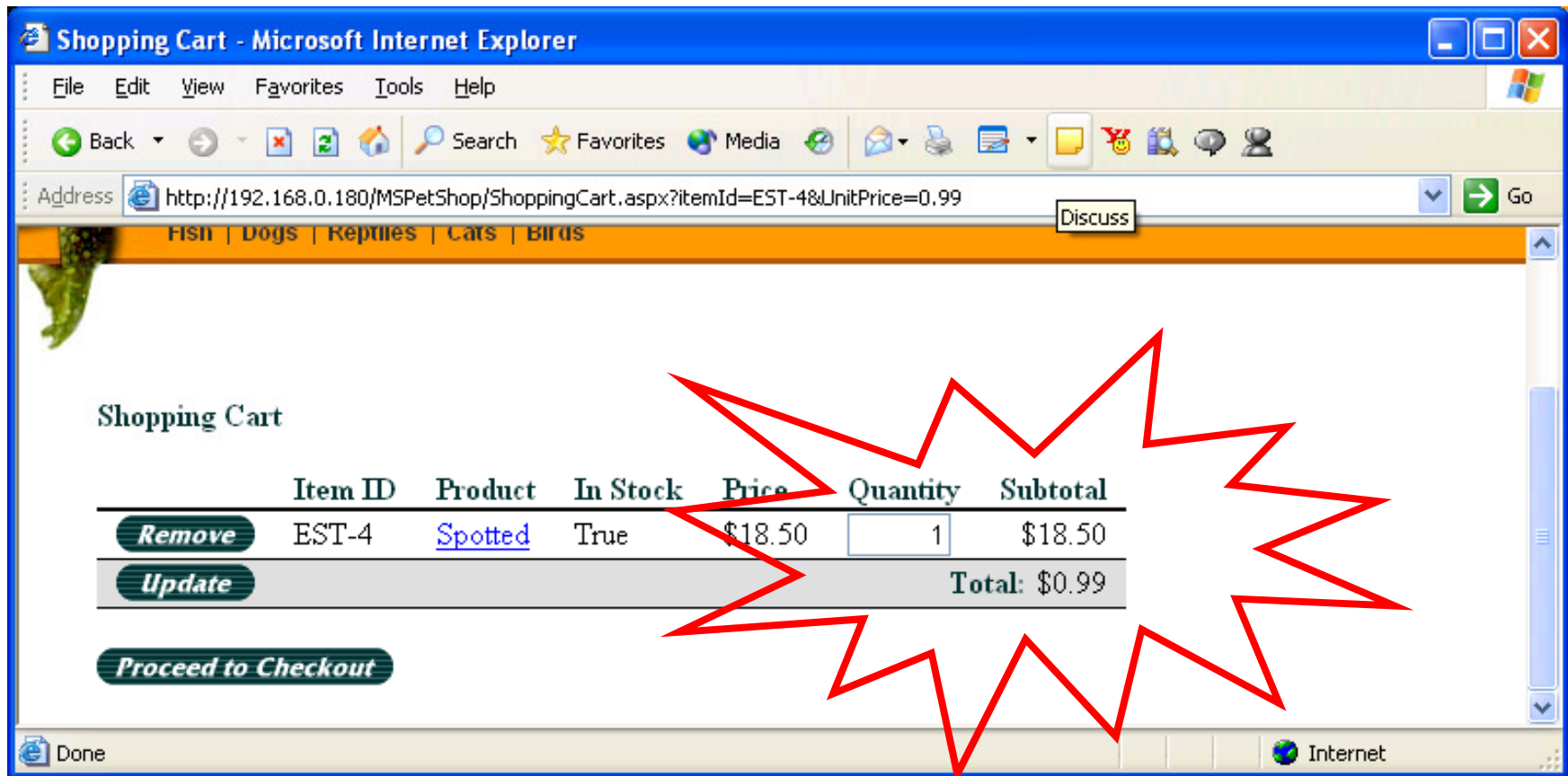
- **Review of faulty application code**

- decimal price = decimal.Parse(Request["UnitPrice"]);
myCart.Add(itemId,price);
- Data stored client side utilized directly in the application business logic
- No verification of pricing (parameter) against server side data

- **Possible mitigation techniques**

- **Do not store sensitive information client side**
 - If you must, encrypt and sign the data
- **Do not trust client data**
 - Verify that user supplied data has not been tampered with

Price Fixing - Demonstration



<http://TARGETIP/MSPetShop/ShoppingCart.aspx?itemId=EST-4&UnitPrice=0.99>

Common Vulnerabilities – Directory Traversal

- **What it is and what does it do**
 - Allow the attacker to download or gain access to files outside of the normal web directory structure
- **Vulnerability identification**
 - Manipulation of parameters with .. And / or \ characters in an attempt to bypass normal directory structures
- **Execution of the vulnerability**
 - Modify the parameters within the URL
 - Alternatively intercept the request and modify hidden fields or modify the content of the HTTP POST

Common Vulnerabilities – Directory Traversal

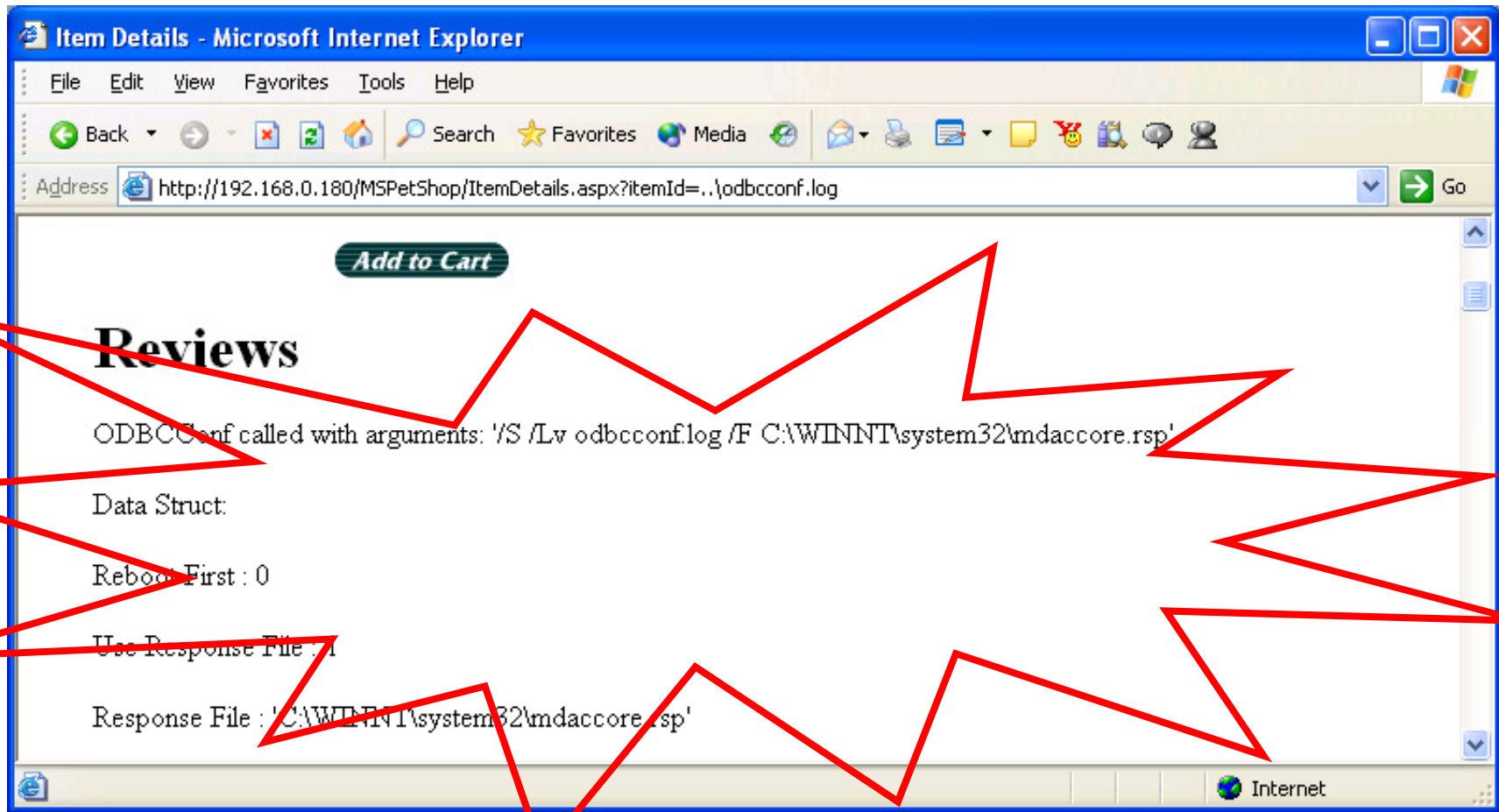
- **Review of faulty application code**

- string path = "c:\\PetStore.review\\"+itemId; using (StreamReader sr = new StreamReader(path))
- Lack of input validation
- File access granted where not specifically required

- **Possible mitigation techniques**

- Validate all input
- Utilize a lower privileged account
- Only grant permissions on specifically required files
- Use “Mappath” method to safely generate a file path

Directory Traversal - Demonstration



<http://TARGETIP/MSPetShop/ItemDetails.aspx?itemId=..\odbcconf.log>

Common Vulnerabilities – Verbose Error Messages

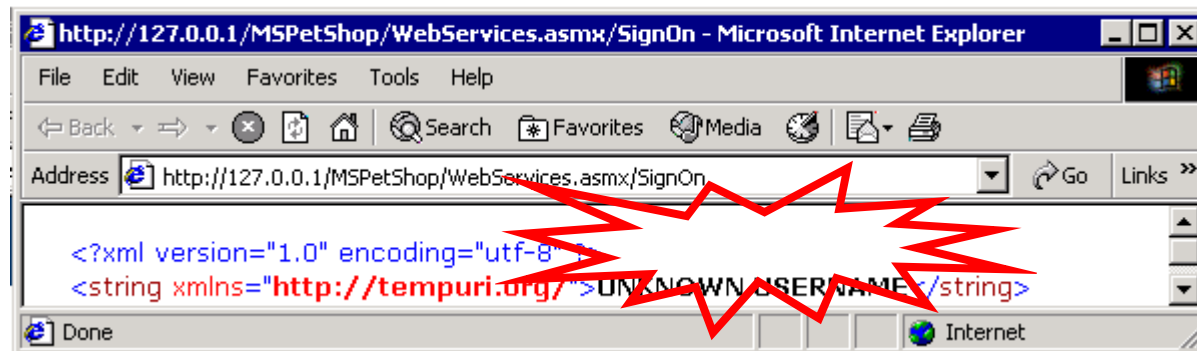
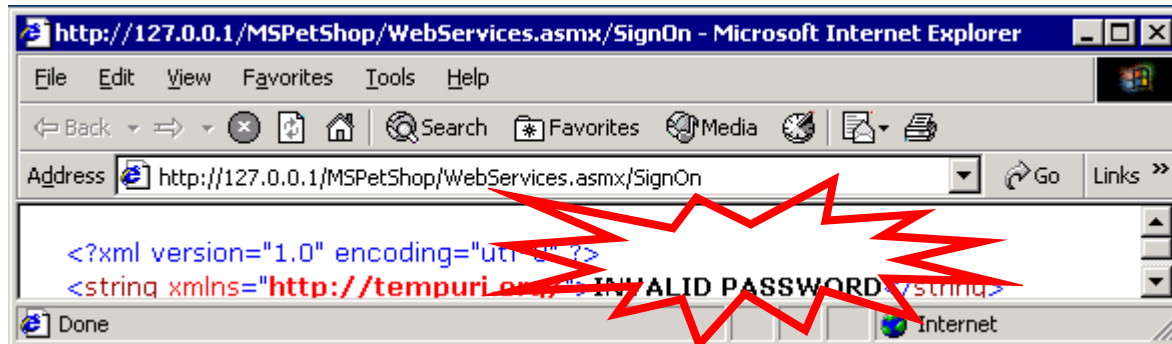
- **What it is and what does it do**
 - Presenting the end user with verbose error messages including potentially sensitive information
 - Utilized to further other attacks against the system
- **Vulnerability identification**
 - No special technique required
 - Review all error messages in detail
- **Execution of the vulnerability**
 - May directly result in an attack vector presenting itself for exploit
 - May leak information that can be used in conjunction with other attack vectors

Common Vulnerabilities – Verbose Error Messages

- **Review of faulty application code**
 - Configuration error allowing the web service to be available for requests
 - Error presents detailed information

- **Possible mitigation techniques**
 - Modify the resulting page to have a limited error message
 - Configure the server so this functionality is not available or remove it all together

Verbose Error Messages - Demonstration



<http://192.168.0.180/MSPetShop/WebServices.asmx?op=SignOn>

Common Vulnerabilities – Information Disclosure

- **What it is and what does it do**

- Default functionality leaks information
- Allows an attacker to gain knowledge about the target environment
- May leak sensitive data

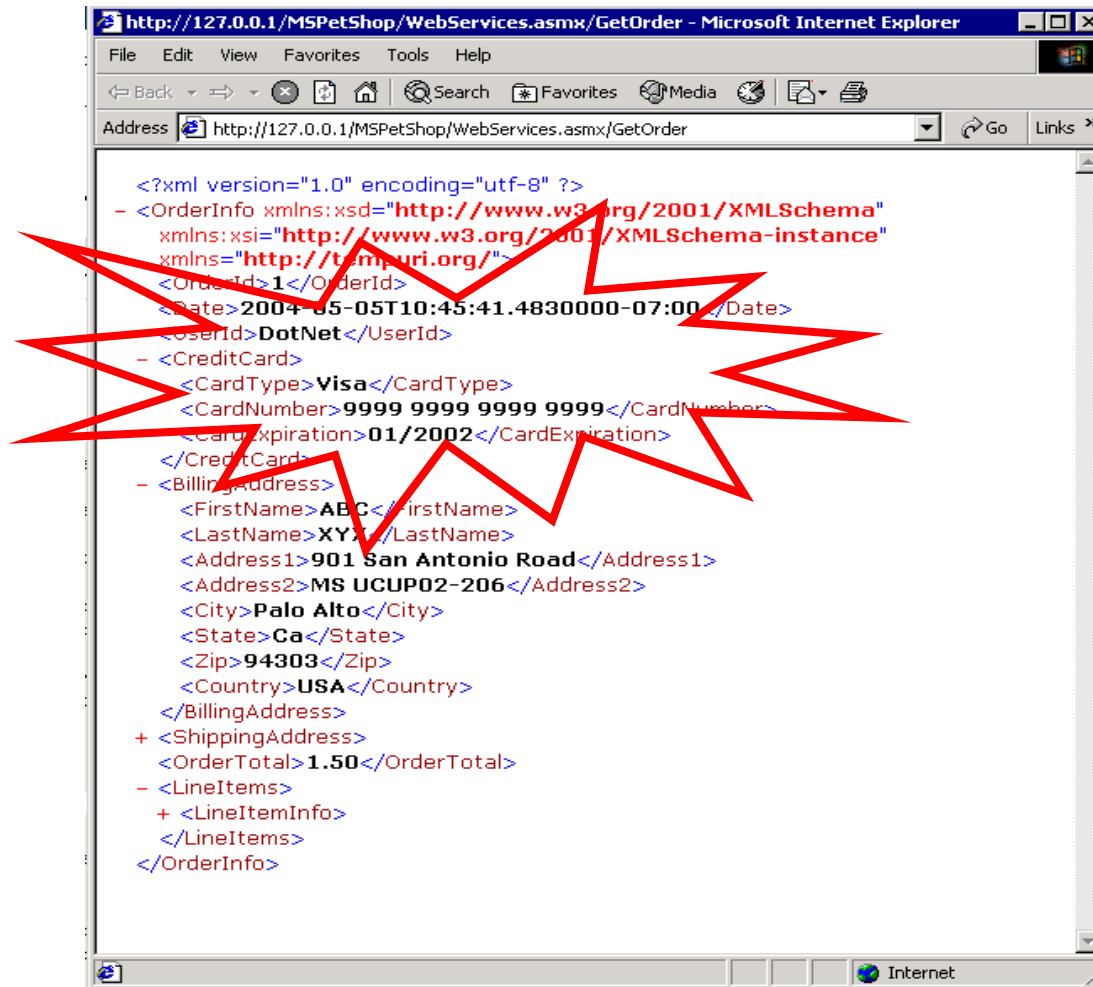
- **Vulnerability identification**

- Subtle differences in information displayed in the browser
- Comments left in production code
- Application functionality that does not require authentication

Common Vulnerabilities – Information Disclosure

- **Execution of the vulnerability**
 - Varies dependant upon the type of information disclosure
 - May be an error scenario
 - May be standard functionality available without authentication
- **Review of faulty application code**
 - Commonly a configuration error
 - May be extra information in comments or in presented data
- **Possible mitigation techniques**
 - Require authentication on all requests
 - Verify presented data can not be used against in other attacks

Information Disclosure - Demonstration



Review - Attacks

- **Cross-site Scripting**
- **SQL Injection**
- **Price Fixing (aka Parameter Tampering)**
- **Directory Traversal**
- **Verbose Error Messages**
- **Information Disclosure**

Review - Defenses

- **Input validation**
- **Output sanitation and encoding**
- **Least privilege accounts and account permissions**
- **Limit error message information**
- **Require authentication and authorization for all requests**
- **Harden the web application removing extraneous content**

Questions

