

Web Application Hacking 101

Common Web Attacks and Defenses

Agenda

- Introduction
- Application Attacks Overview
- Attack Methodologies
- Demonstration Platform Details
- Demo of Common Vulnerabilities
 - Cross-site Scripting
 - SQL Injection
 - Price Fixing (Parameter Tampering)
 - Directory Traversal
 - Verbose Error Messages
 - Information Disclosure
- Review
- Questions

Introduction

- txs@0x90.org
 - Active member of 0x90.org “Digital Think Tank” and “Source of Proper Internet Villainy” (www.0x90.org)
 - Founding member of the GhettoHackers, GHI (www.ghettohackers.net)
 - Professional security consultant to the U.S. Government, Financial Institutions, Fortune 500 Companies, and anyone willing to pay cash.
 - Primary focus is application penetration testing and analysis. In other words I break stuff.
 - 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.

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 defenses, 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: @Stake/Symantec WebProxy, Paros, 0x90.org Sake

Attack Methodology: Target

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

Attack Methodology: Attack

Ok enough of this chit-chat. Show me how to break something!

Demonstration Platform Details

- “Microsoft .Net PetShop 3.0” is Microsoft’s implementation of a .NET 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 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

Vulnerability 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

Verbose Error Messages

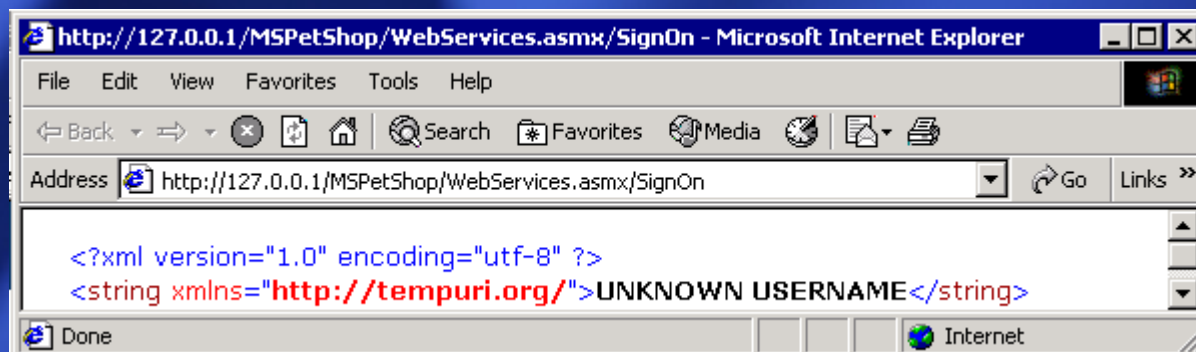
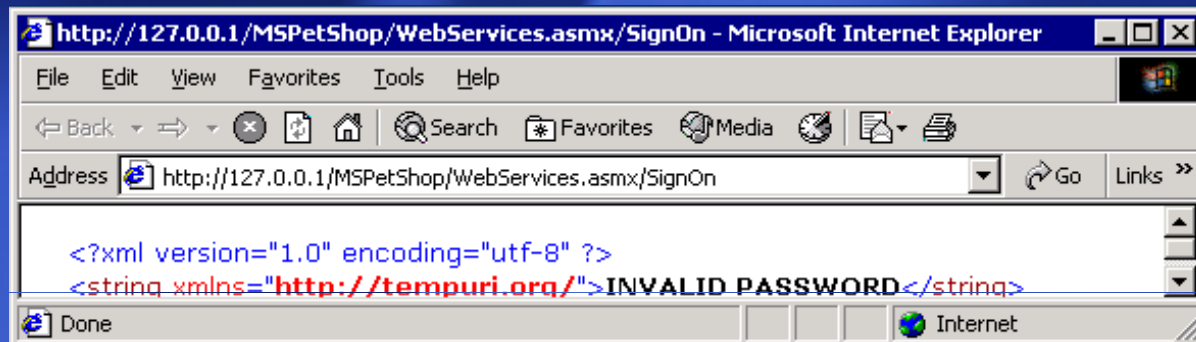
- What it is and what does it do
 - Presenting the end user with verbose error messages including potentially sensitive information
 - Utilized to further other attacks against the system
- Vulnerability identification
 - No special technique required
 - Review all error messages in detail
- Execution 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

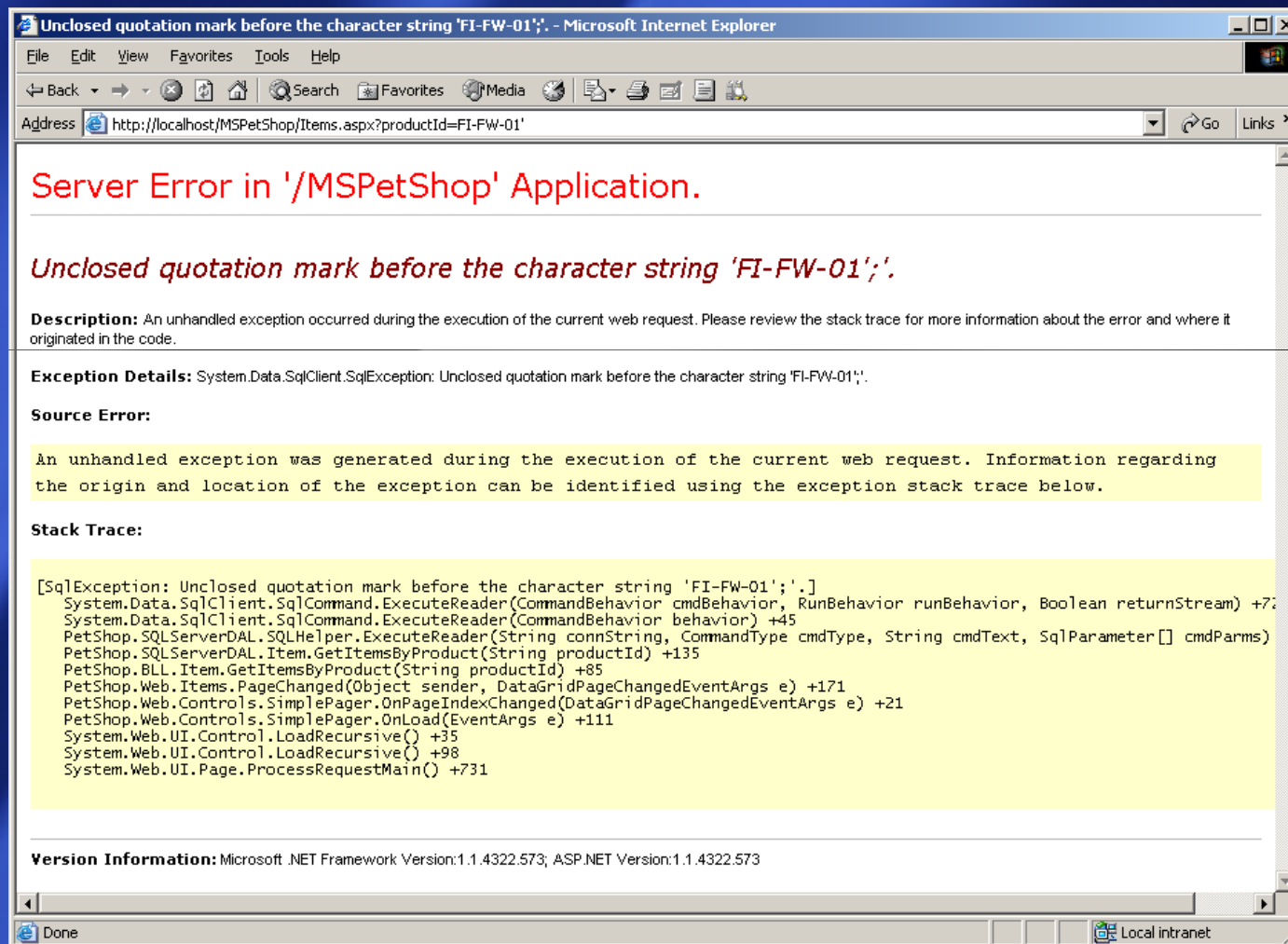
Verbose Error Messages

- 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



Verbose Error Messages Demonstration



The screenshot shows a Microsoft Internet Explorer browser window with the title "Unclosed quotation mark before the character string 'FI-FW-01';. - Microsoft Internet Explorer". The address bar contains the URL "http://localhost/MSPetShop/Items.aspx?productId=FI-FW-01". The main content area displays a red error message: "Server Error in '/MSPetShop' Application." followed by the error text "Unclosed quotation mark before the character string 'FI-FW-01';." in red. Below this, a "Description" section explains that an unhandled exception occurred. The "Exception Details" section shows the error type as "System.Data.SqlClient.SqlException: Unclosed quotation mark before the character string 'FI-FW-01';." The "Source Error" section contains a yellow-highlighted message: "An unhandled exception was generated during the execution of the current web request. Information regarding the origin and location of the exception can be identified using the exception stack trace below." The "Stack Trace" section also features a yellow highlight and lists the following call stack entries: [SqlException: Unclosed quotation mark before the character string 'FI-FW-01';.] System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream) +7; System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior) +45 PetShop.SQLServerDAL.SQLiteHelper.ExecuteReader(String connString, CommandType cmdType, String cmdText, SqlParameter[] cmdParms) PetShop.SQLServerDAL.Item.GetItemsByProduct(String productId) +135 PetShop.BLL.Item.GetItemsByProduct(String productId) +85 PetShop.Web.Items.PageChanged(Object sender, DataGridPageChangedEventArgs e) +171 PetShop.Web.Controls.SimplePager.OnPageIndexChanged(DataGridPageChangedEventArgs e) +21 PetShop.Web.Controls.SimplePager.OnLoad(EventArgs e) +111 System.Web.UI.Control.LoadRecursive() +35 System.Web.UI.Control.LoadRecursive() +98 System.Web.UI.Page.ProcessRequestMain() +731 At the bottom, the "Version Information" section states: "Microsoft .NET Framework Version:1.1.4322.573; ASP.NET Version:1.1.4322.573". The browser's status bar at the bottom shows "Done" and "Local intranet".

Common Vulnerabilities

Information Disclosure

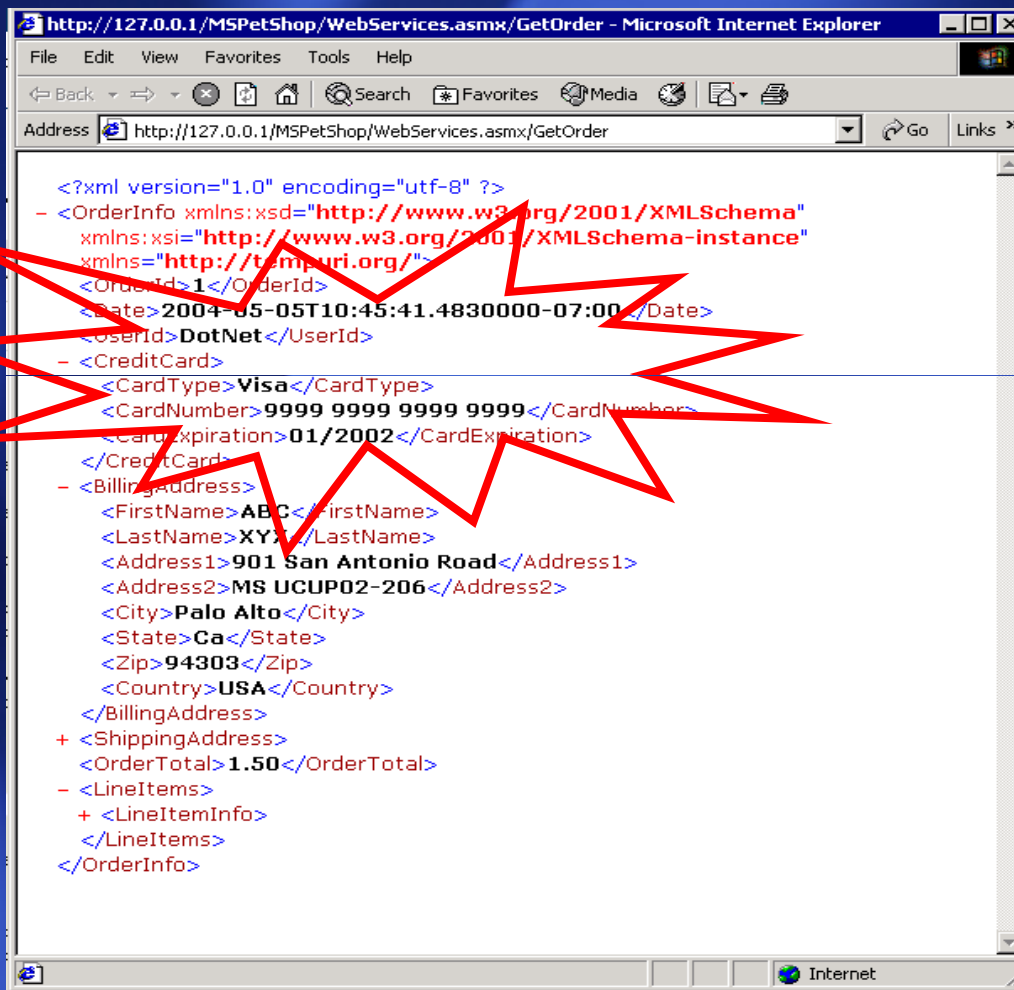
- What it is and what does it do
 - Default functionality leaks information
 - Allows an attacker to gain knowledge about the target environment
 - May leak sensitive data
- Vulnerability identification
 - Subtle differences in information displayed in the browser
 - Comments left in production code
 - 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 in other attacks

Information Disclosure Demonstration



```
<?xml version="1.0" encoding="utf-8" ?>
- <OrderInfo xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://tempuri.org/">
  <OrderId>1</OrderId>
  <Date>2004-05-05T10:45:41.4830000-07:00</Date>
  <UserId>DotNet</UserId>
  - <CreditCard>
    <CardType>Visa</CardType>
    <CardNumber>9999 9999 9999 9999</CardNumber>
    <CardExpiration>01/2002</CardExpiration>
  </CreditCard>
  - <BillingAddress>
    <FirstName>ABC</FirstName>
    <LastName>XYZ</LastName>
    <Address1>901 San Antonio Road</Address1>
    <Address2>MS UCUP02-206</Address2>
    <City>Palo Alto</City>
    <State>Ca</State>
    <Zip>94303</Zip>
    <Country>USA</Country>
  </BillingAddress>
  + <ShippingAddress>
  <OrderTotal>1.50</OrderTotal>
  - <LineItems>
    + <LineItemInfo>
  </LineItems>
</OrderInfo>
```

Common Vulnerabilities

Directory Traversal

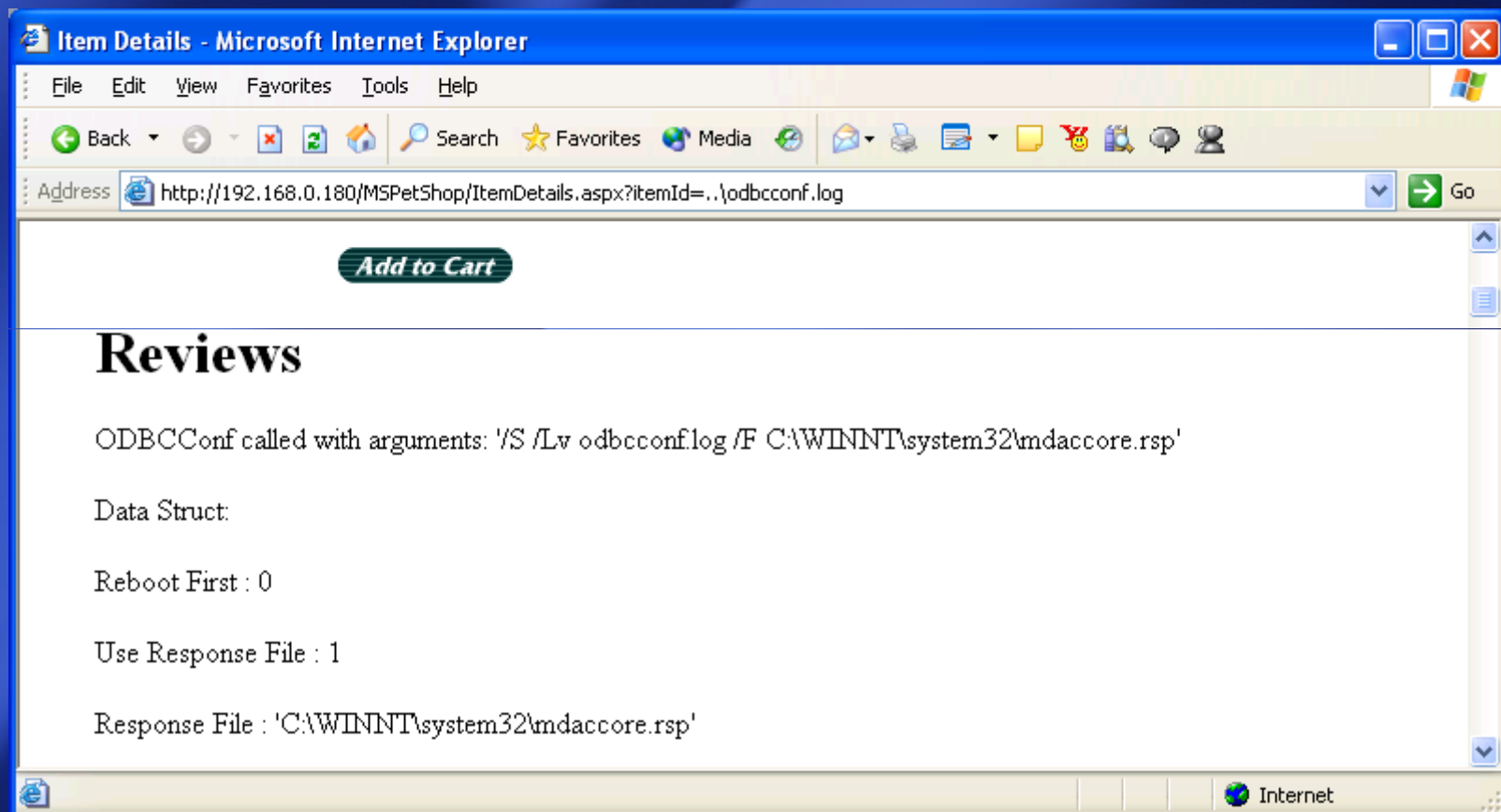
- What it is and what does it do
 - Allow the attacker to download or gain access to files outside of the normal web directory structure
- Vulnerability identification
 - Manipulation of parameters with .. And / or \ characters in an attempt to bypass normal directory structures
- Execution of the vulnerability
 - 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



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

Shopping Cart - Microsoft Internet Explorer

Address: <http://192.168.0.180/MSPetShop/ShoppingCart.aspx?itemId=EST-4&UnitPrice=0.99>

Fish | Dogs | Reptiles | Cats | Birds

Shopping Cart

	Item ID	Product	In Stock	Price	Quantity	Subtotal
Remove	EST-4	Spotted	True	\$18.50	<input type="text" value="1"/>	\$18.50
Update						Total: \$0.99

[Proceed to Checkout](#)

Done Internet

Common Vulnerabilities

Cross Site Scripting (XSS)

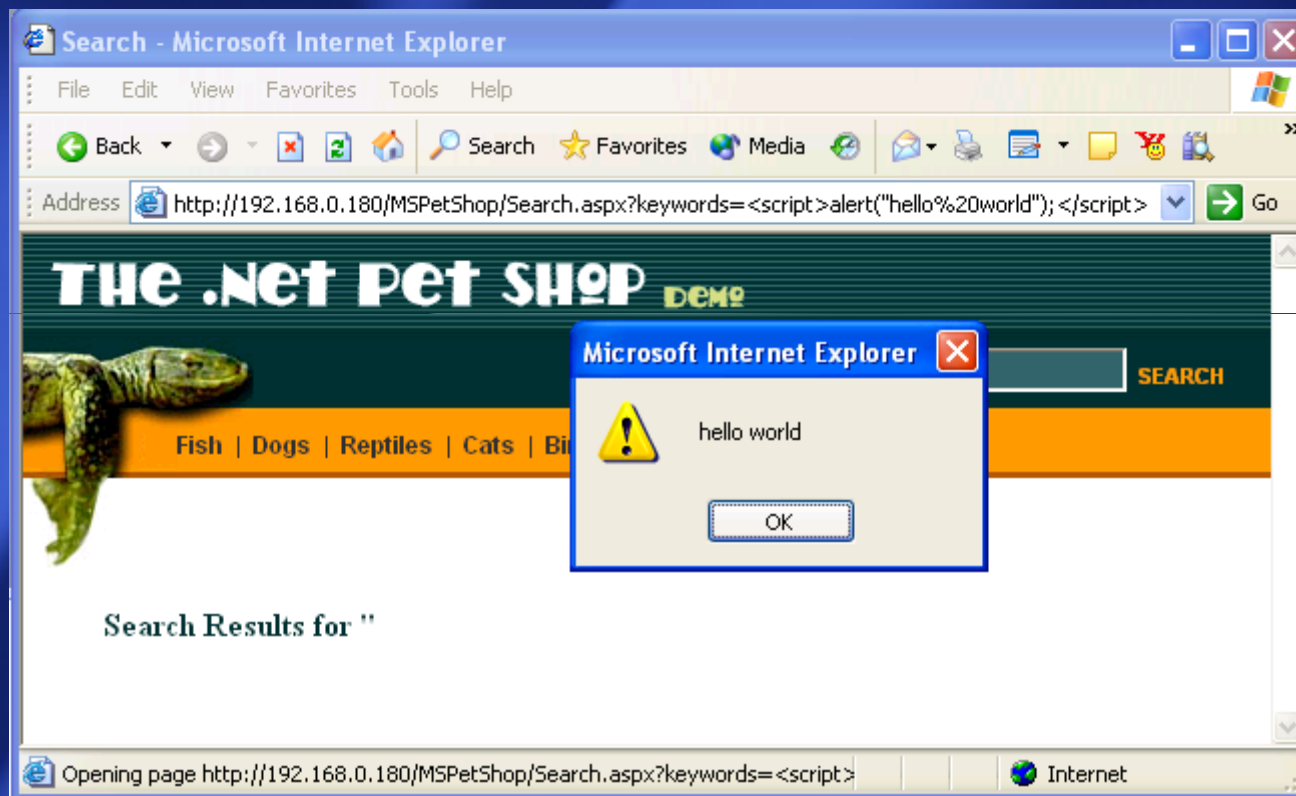
- What it is and what does it do
 - XSS is the injection of code or script into a web page that is then displayed to a third party executing the script in the context of their web browser.
- Vulnerability identification
 - Look for anywhere where client supplied data is redisplayed to the end user and 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



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

Items - Microsoft Internet Explorer

Address http://192.168.0.180/MSPetShop/Items.aspx?page=0&productId=FI-FW-01'%20union%20select%20*%20from%20credentials;-- Go

ACID

Item ID	Name	Price	
ACID	ACID	\$9.00	Add to Cart
aUser	MyNotSoSecretPasswor	\$9.00	Add to Cart
DotNet	DotNet	\$9.00	Add to Cart
EST-4	Spotted	\$18.50	Add to Cart

[More](#)

Internet

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

