

INNOV-09

# How to Keep Hackers Out of your Web Application

Michael Solomon, CISSP PMP CISM

Solomon Consulting Inc.

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



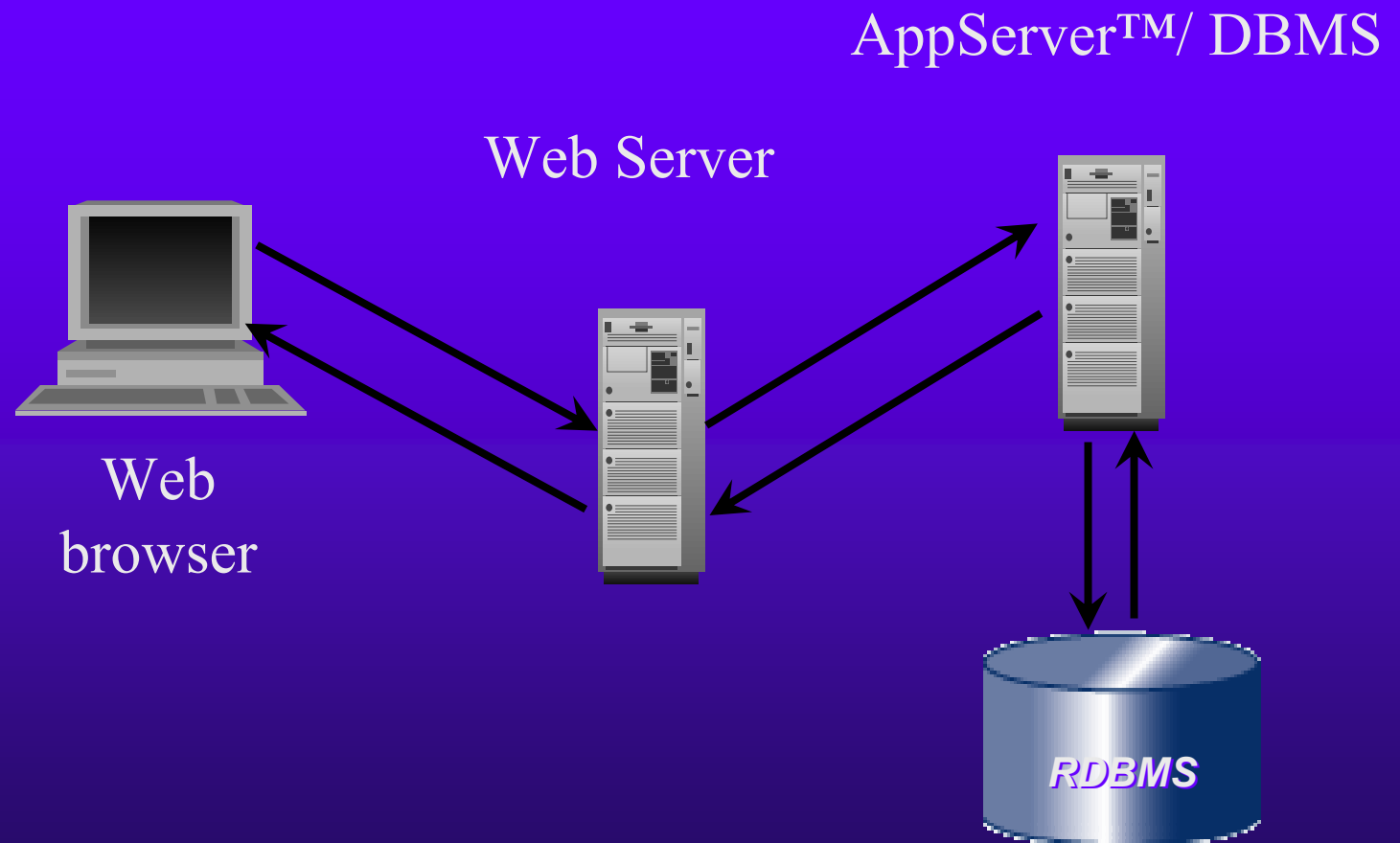
PROGRESS SOFTWARE  
**Exchange**  
2005



# What is a Web Application?

- ◆ Any access to your data via the Internet
  - Generally uses standard web browser
  - Easy to find and connect
- ◆ Advantages
  - No special end user software requirements
  - Connect from anywhere
  - Convenient for users
  - Easy to upgrade software (internal)

# Basic Web Application Architecture



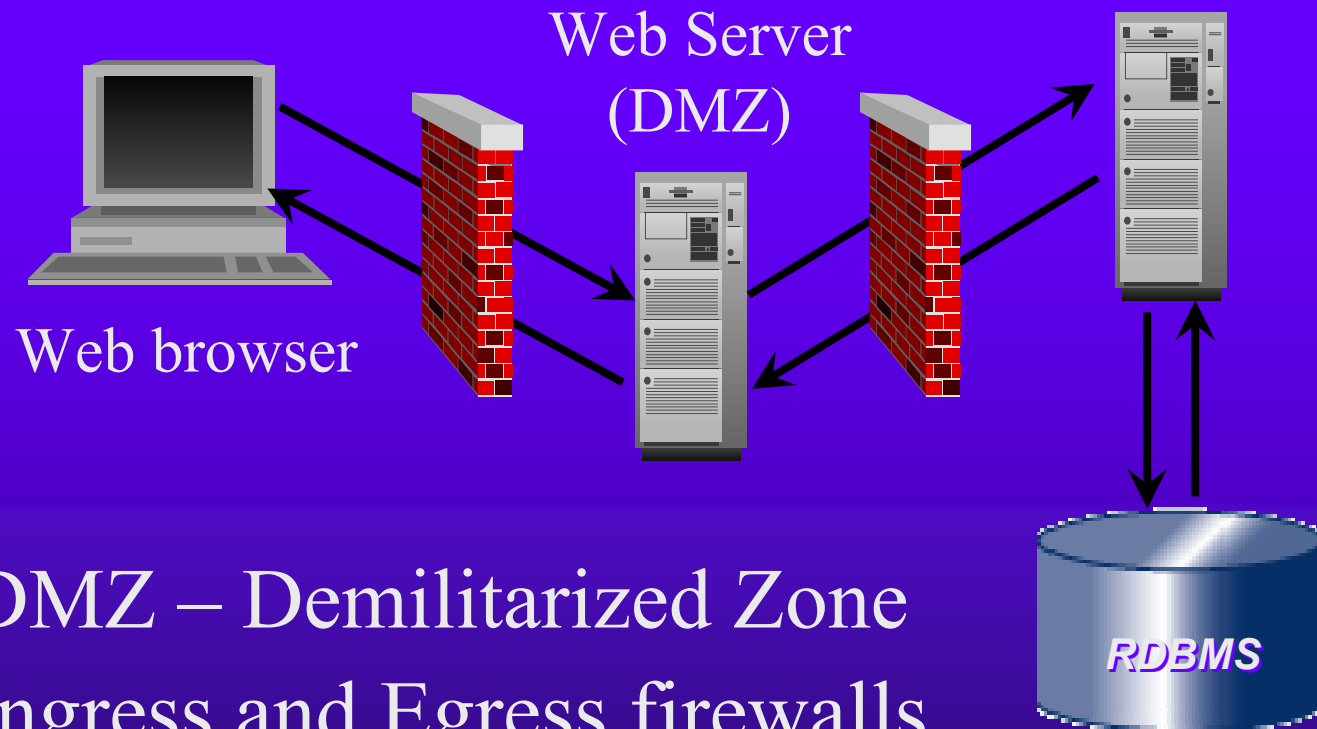


# Web Application Security Issues

- ◆ Most web applications use standard ports
  - HTTP uses port 80
  - HTTPS (SSL) uses port 443
- ◆ Easy access through the exterior firewall
  - Connect to a web app and you are “inside” the company’s system
  - Hopefully, there is at least one more firewall between the web server and the internal network
- ◆ Many weaknesses are known, but they still persist
  - We keep making the same mistakes over and over
  - See the SANS Top 20 Internet Vulnerabilities List
    - [www.sans.org/top20](http://www.sans.org/top20)

# Perimeter Defense

AppServer™/DBMS



- ◆ DMZ – Demilitarized Zone
- ◆ Ingress and Egress firewalls
- ◆ Still not secure enough



# What are hackers after?

- ◆ Regardless of the motivation, there are three basic goals of attacks on web applications
  - Disclosure of confidential data
    - Medical records/Financial information
    - Identity theft
  - Modification of important data
    - Grades
    - Medical/Financial information
  - Interruption of service
    - Denial of Service (DoS)



# How to stop hackers

- ◆ Realistically, you can't
  - A motivated attacker will almost always find a way to attack the intended target
- ◆ But, you can dramatically reduce the likelihood that an attack on your system will be successful
  - Know how attackers carry out attacks
  - Place as many roadblocks in their way as you can
  - Put “The Club” on your system
    - Don't be “low hanging fruit”
- ◆ Let's look at how hackers plan attacks



# The path to a successful attack

Most successful attacks follow these steps:

1. Collect information on target
  - ◆ Profile the system
  - ◆ Profile the application
2. Analyze information to determine target's weaknesses
3. Launch attacks against known weaknesses



## Reconnaissance – collecting information

- ◆ The first step in a successful attack is to profile the target
  - A good target profile can uncover many potential weaknesses
- ◆ Examples:
  - Un-patched software
  - Open ports
  - Weak authentication
  - Poor security on sensitive data



# Profiling utilities and methods

- ◆ Nmap – Port scanner
  - [www.insecureorg/nmap](http://www.insecureorg/nmap)
- ◆ Netcat – All-purpose network sockets utility
  - [www.securityfocus.com/tools/137](http://www.securityfocus.com/tools/137)
  - <http://www.securityfocus.com/tools/139/scoreit>
- ◆ Nikto – Vulnerability scanner
  - <http://www.cirt.net/code/nikto.shtml>
- ◆ Nessus – Vulnerability scanner
  - <http://www.nessus.org/>



# Pay attention to system scans

- ◆ Generally, scanning is not illegal
- ◆ A system scan may be the first step in an attack on your system
  - Keep records of scanning activity
  - You can use scanning information to help identify subsequent attackers
- ◆ Don't overreact to scanning activity
  - Most scans will not result in future attacks
  - The fewer holes that scans turn up, the better your chances of avoiding an attack



## Analyze information to uncover weaknesses

- ◆ Don't move to this step too soon!
  - Be patient and collect as much information as you can about your target
- ◆ Once you have collected all the information you can on your intended target, start looking for easy hits
- ◆ Depending on your attack motives, choose the most effective attack



## Ask for help – Social Engineering

- ◆ Social engineering can be the easiest way into a system
  - Social engineering – the act of convincing an authorized user to perform actions that you are not authorized to carry out
- ◆ Never underestimate the power of social engineering attacks
- ◆ People are generally willing to help



# Sources for known vulnerabilities

- ◆ Only a few of the many resources available
  - [www.packetstormsecurity.org](http://www.packetstormsecurity.org)
  - [www.securityfocus.com](http://www.securityfocus.com)
  - [www.cgisecurity.com](http://www.cgisecurity.com)
  - [www.owasp.org](http://www.owasp.org)
  - [www.sans.org](http://www.sans.org)



# Common attacks

- ◆ Input validation
  - Targets any modifiable data sent to the server
- ◆ SQL injection (works with 4GL, too)
  - Submitting cleverly crafted query arguments to provide more or different data than was intended
- ◆ Cross-site scripting
  - A method of causing authorized users to provide data to unauthorized recipients



# Common attacks

- ◆ Stored-data attacks

- Taking advantage of, or stealing, any data stored on a client machine

- ◆ Session attacks

- Hijacking a session and assuming another identity

- ◆ XML attacks

- Combination of input validation and SQL injection attacks, but using XML as a transport format



## Launch the attack

- ◆ Once you have chosen the appropriate attack methods, launch the attack
- ◆ Timing may be important
  - Choose a time where immediate response is unlikely
  - Weekends and holiday often result in smaller support staffing



# Input validation attacks

- ◆ The #1 most common (and critical) web application vulnerability (OWASP)
- ◆ Try to send data the application does not expect to cause:
  - Data disclosure or modification
  - System compromise
  - Denial of Service (DoS)
  - Arbitrary command execution
- ◆ For example, a common method is to replace some characters with URL encoded characters
  - <http://system/scripts/..%c0%af..%c0%afdir/cmd>
  - Becomes: <http://system/scripts/../../dir/cmd>
  - You just convinced the web service to execute a command!



# Input validation attacks

- ◆ How to protect your OpenEdge™ application
  - Validate ALL input data, regardless of source
    - Validate on the server, even if client-side validation has already been applied
      - Client-side validation enhances usability
      - Service-side validation enhances security
  - Validate ALL output data as well
  - Decode all characters BEFORE applying validation rules
    - Process is called ‘canonicalization’



# Query injection attacks

- ◆ The goal is to extract or modify data that is not supposed to be available to you
- ◆ Alter input data to prematurely terminate or add query arguments
  - Not only SQL; can work in 4GL
- ◆ For example, you could enter the following data in the “SearchFor” field: **1" or "2" = "2**
  - `QUERY-PREPARE(“FOR EACH item WHERE ItemName CONTAINS ~” + VALUE(cSearchFor) + “~”)`.



# Query injection attacks

- ◆ How OpenEdge applications are vulnerable
  - Dynamic queries
  - Query objects
  - SmartDataObjects
- ◆ Any use of the QUERY-PREPARE method
- ◆ Solution:
  - Pre-process ALL query string components!



# Cross-site scripting attacks

- ◆ An attack that doesn't target the server, but other users
- ◆ Cross-site scripting (XSS) attacks cause other users of a web application to execute code you specify
- ◆ For example, enter this in the "SearchFor" field:
  - `<script>alert('Hello, world');</script>` in the SearchFor field.
  - If a web application allows arbitrary script code, you could insert any script into input fields
- ◆ If the previous string value doesn't work, try this:
  - `%3Cscript%3Ealert('Hello%2C%20world')%3B%3C/script%3E`
  - This approach combines input validation and XSS attacks



# Cross-site scripting attacks

- ◆ How to protect your OpenEdge application
  - Same approach as with input validation attacks
  - Ensure all input (and output) are validated on the server



# Stored-data attacks

- ◆ Many web applications store data on the client machine
  - Cookies
  - Favorites
- ◆ Changing stored data can cause unexpected results
  - Force initial behavior
  - Send authentication or other sensitive data to another location without the user's knowledge
- ◆ Sometimes called 'data poisoning'



# Stored-data attacks

- ◆ How to protect your OpenEdge application
  - Start by validating all input (and output) data
  - Limit data stored on (or transmitted to) client machines
  - Eliminate sensitive data stored or transmitted “in the clear”
  - Carefully choose the right encryption method for storing sensitive data on the client



# Session attacks

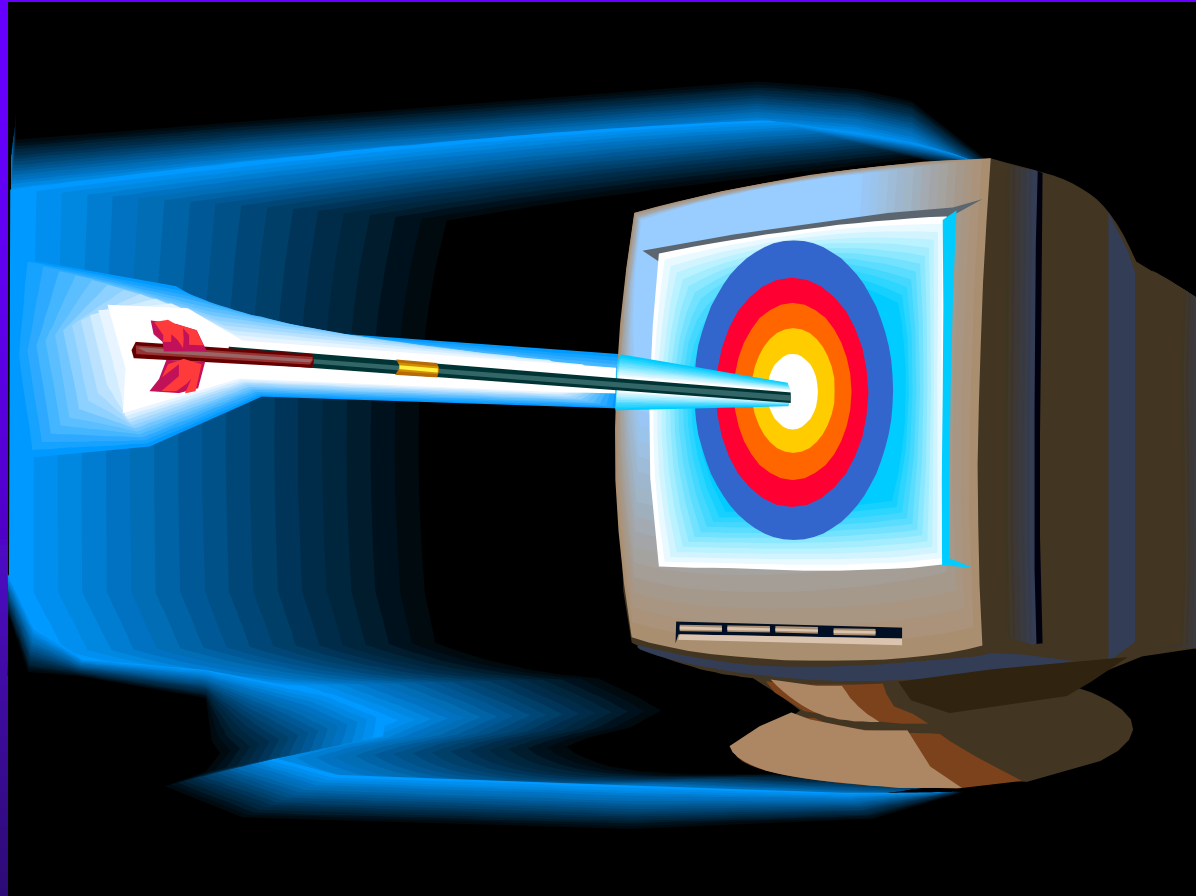
- ◆ Session attacks exploit poorly implemented session management
  - HTTP was designed to operate using discrete requests – no concept of sessions
  - The application must manage session information
- ◆ Attackers can attempt to alter or hijack a session by manipulating session tracking variables
  - Often URL arguments or cookies
  - Session ID could be
    - Sequential
    - Time/date based
    - Pseudorandom
    - Constructed



# Session attacks

- ◆ How to protect your OpenEdge application
  - Develop a sound session management system
  - Store session information in a DB table
  - Use a unique id to identify a session
    - The original client session id works
    - Associate the session id with user and timestamp information
  - Ensure sessions are only valid for a short period of time and tied to a single IP address

# Web Application Attack Demo





# The OWASP Top 10 Web Application Vulnerabilities List

- ◆ For more information
  - The OWASP web site [www.owasp.org](http://www.owasp.org)
- 1. Unvalidated Input
- 2. Broken Access Control
- 3. Broken Authentication and Session Management
- 4. Cross Site Scripting (XSS) Flaws
- 5. Buffer Overflows



# The OWASP Top 10 Web Application Vulnerabilities List

6. Injection Flaws
7. Improper Error Handling
8. Insecure Storage
9. Denial of Service
10. Insecure Configuration Management



# Web application hardening checklist

- ◆ Patch your software!
  - OS, firewall, web server, AppServer, DBMS
- ◆ Assess your system for the OWASP Top 10 web application vulnerabilities
  - <http://www.owasp.org/documentation/topten.html>
- ◆ Assess your system for the SANS Top 20 Internet Security Vulnerabilities
  - [www.sans.org/top20](http://www.sans.org/top20)



# For More Information

- ◆ Solomon Consulting Inc.
  - [www.solomonconsulting.com](http://www.solomonconsulting.com)
- ◆ The SANS Institute
  - [www.sans.org](http://www.sans.org)
- ◆ Web-related vulnerabilities information
  - [www.cgisecurity.org](http://www.cgisecurity.org)
- ◆ The WWW Security FAQ
  - [www.w3.org/Security/Faq/](http://www.w3.org/Security/Faq/)
- ◆ Open Web Application Security Project
  - [www.owasp.org](http://www.owasp.org)