



# OWASP and the OWASP Top 10 (2007 Update)

Sebastien Deleersnyder  
OWASP BE Chapter Leader

**OWASP**

CISSP, CISA, CISM  
AppSec consultant  
seba@deleersnyder.eu

Copyright © 2007 - The OWASP Foundation  
This work is available under the Creative Commons SA 2.5 license

**The OWASP Foundation**  
<http://www.owasp.org>

# Agenda

- OWASP?
- OWASP Top 10, v2007RC1
- Belgium Chapter

# Agenda

- OWASP?
- OWASP Top 10, v2007RC1
- Belgium Chapter

# OWASP

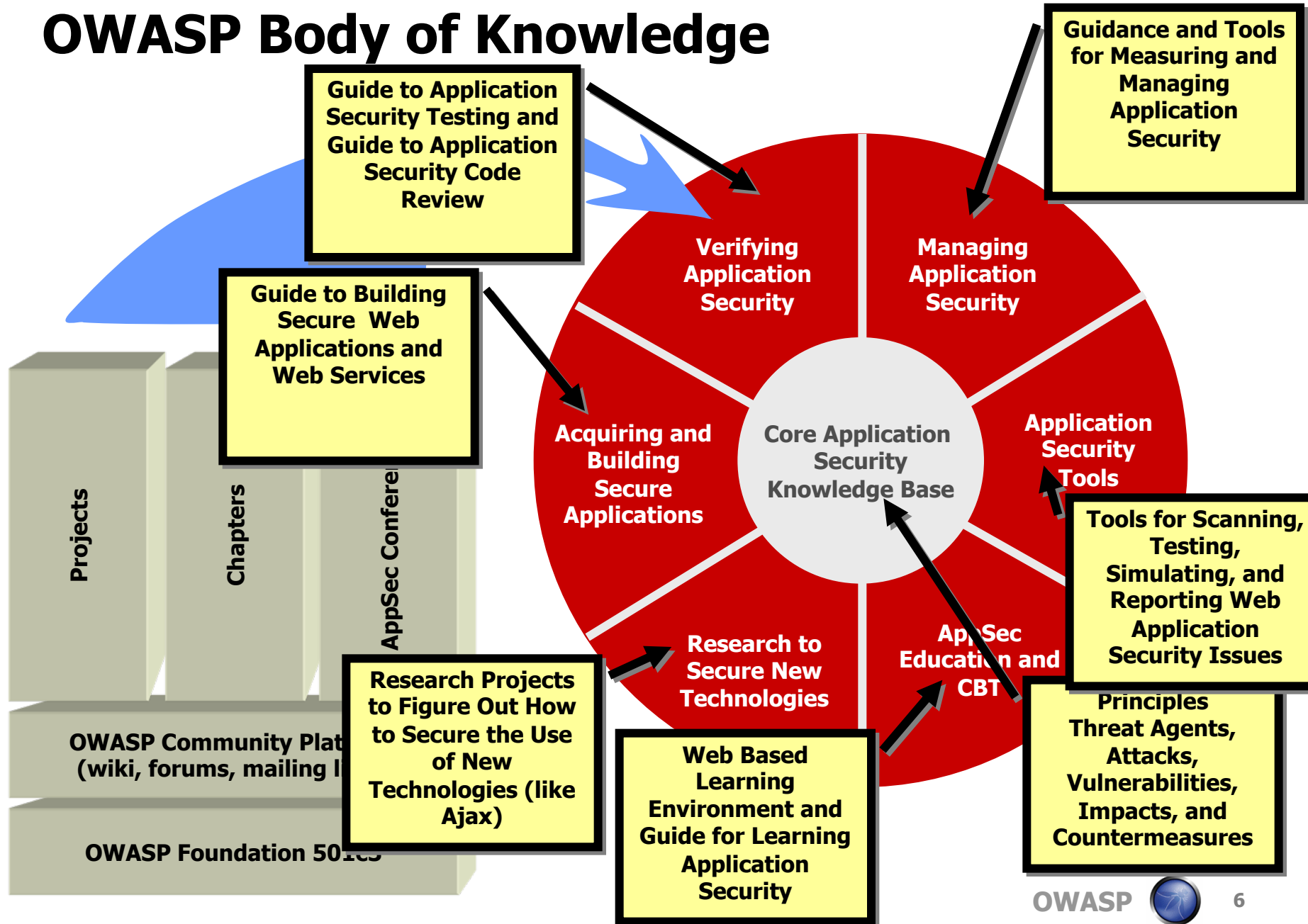
- The Open Web Application Security Project (OWASP)
- International not-for-profit charitable Open Source organization funded primarily by volunteers time, OWASP Memberships, and OWASP Conference fees
- Participation in OWASP is free and open to all

# OWASP Mission

- To find and fight the causes of insecure software



# OWASP Body of Knowledge



# www.owasp.org (our wiki)



The screenshot shows the OWASP website's main page. At the top right, there is a 'Log in / create account' link. Below the navigation tabs (article, discussion, view source, history), the 'Main Page' title is displayed. A sponsored advertisement for Watchfire's AppScan 7.0 Solution is featured. The central 'Welcome to OWASP' section includes a list of resources such as Guide, Top Ten, WebGoat, CLASP, WebScarab, Contracting, Testing, and Code Review. A sidebar on the left contains navigation and reference links. The main content area includes an 'OWASP Overview' section, a 'Featured Story' about the Spring Of Code awarding grants, and an 'OWASP Conferences' section for the AppSec in Italy event. A world map in the 'OWASP Community' section shows global member locations.

Log in / create account

article discussion view source history

## Main Page

Download Watchfire's AppScan™ 7.0 Solution Today!

Sponsored advertisement. OWASP does not endorse commercial products or services.

### Welcome to OWASP

the free and open application security community

- Guide
- Top Ten
- WebGoat
- CLASP
- WebScarab
- Contracting
- Testing
- Code Review
- More...

About - Searching - Editing - New Article - OWASP Categories

#### OWASP Overview

The Open Web Application Security Project (OWASP) is dedicated to finding and fighting the causes of insecure software. Everything here is free and open source. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work. Participation in OWASP is free and open to all.

- Join webappsec! The OWASP way!
- Get Started Find out more
- Contact OWASP owasp@owasp.org
- Become a Member Support our efforts

#### Featured Story

##### OWASP Spring Of Code Awarding \$150,000 in Grants - Apply NOW



Last Fall, the OWASP Foundation gave \$35,000 in grants to worthy application security projects. This Spring, we're using current membership fees and profits from past conferences to fund a bigger round of projects.

As a special offer, 100% of all membership fees collected during the Spring of Code application period will be added to the total Spring of Code dollar amount awarded.

All applications are due by March 30th. The OWASP Spring of Code is not connected to the Google Summer of Code.

#### OWASP Conferences


##### Registration Open for OWASP AppSec in Italy May 15-17

Join us for our 6th AppSec Conference May 15-17 in Milan, Italy. Microsoft will be presenting "The Benefits of the SDL initiative to Microsoft and its Customers" and there will be expert talks on Web Services Security, Securing AJAX, the Microsoft Secure Development Lifecycle, all the new OWASP projects, and much more.

Since we're a nonprofit we can make OWASP AppSec an incredible bargain. If you act by April 15, registration is just 400 Euros, 360 Euros for OWASP Members, and only 200 Euros for Students. You can attend one of 3 full-day training sessions on the 15th, and the main conference is two full days of presentations, panels, and discussion on the 16th and 17th.

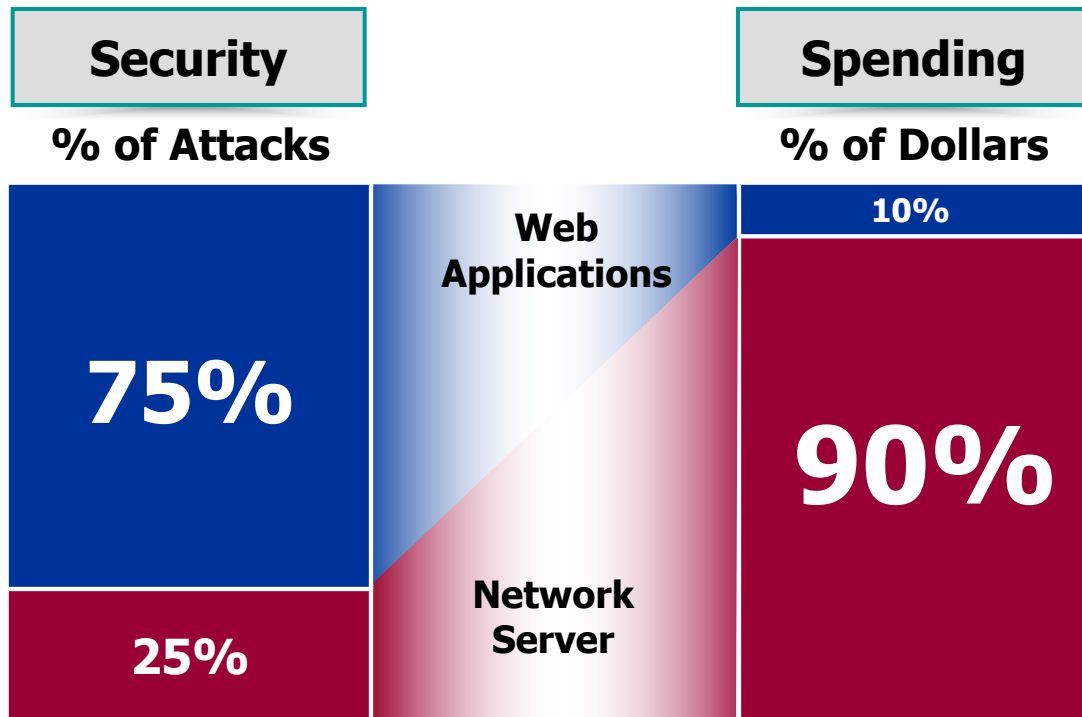
You can read all the details and then register online.

#### OWASP Community (add)



# Why OWASP?

## Attacks Shift Towards Application Layer

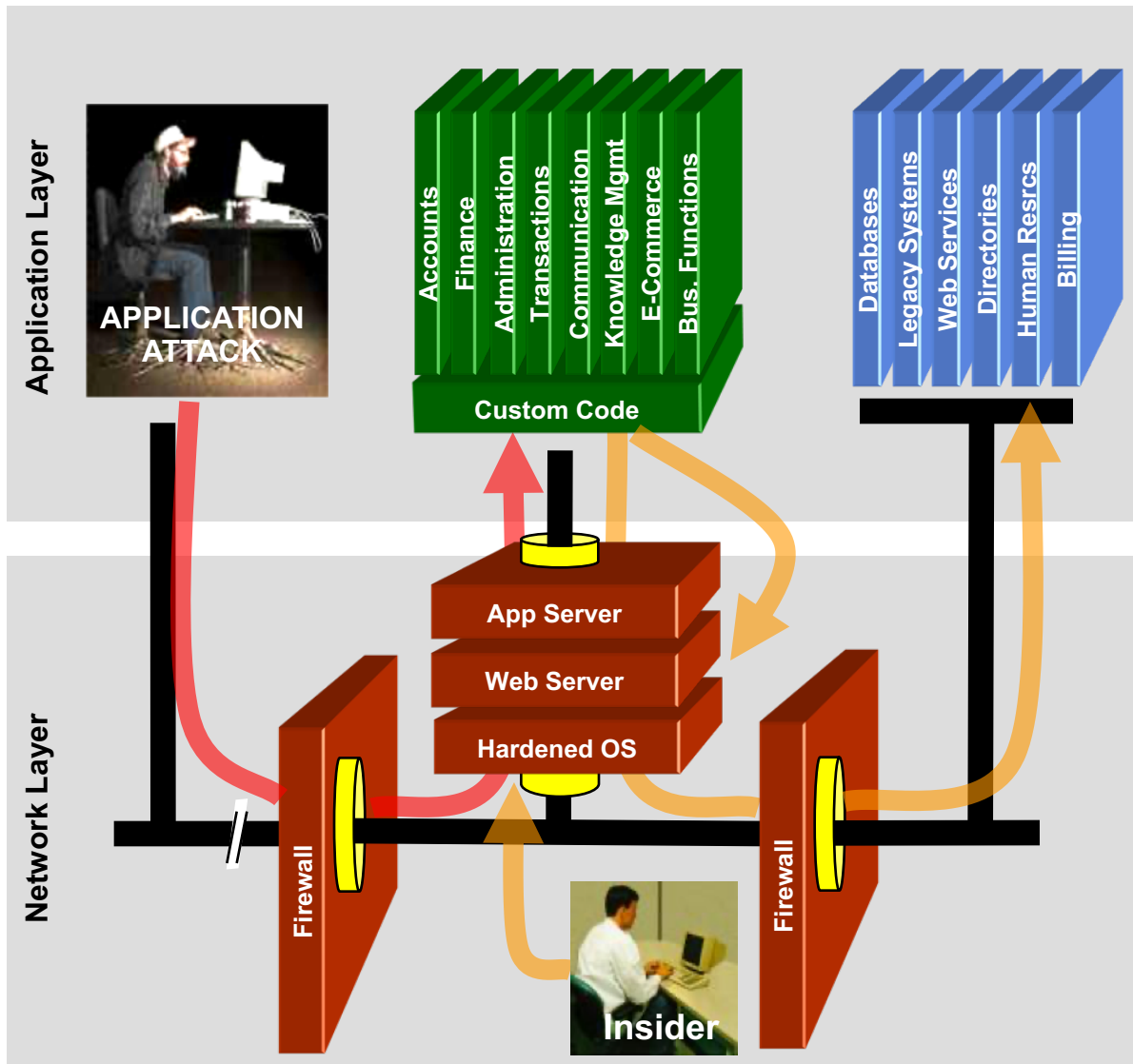


**2/3** of All Web Applications Are Vulnerable

Gartner



# Problem Illustration



## Application Layer

- ▶ Attacker sends attacks inside valid HTTP requests
- ▶ Your custom code is tricked into doing something it should not
- ▶ Security requires software development expertise, not signatures

## Network Layer

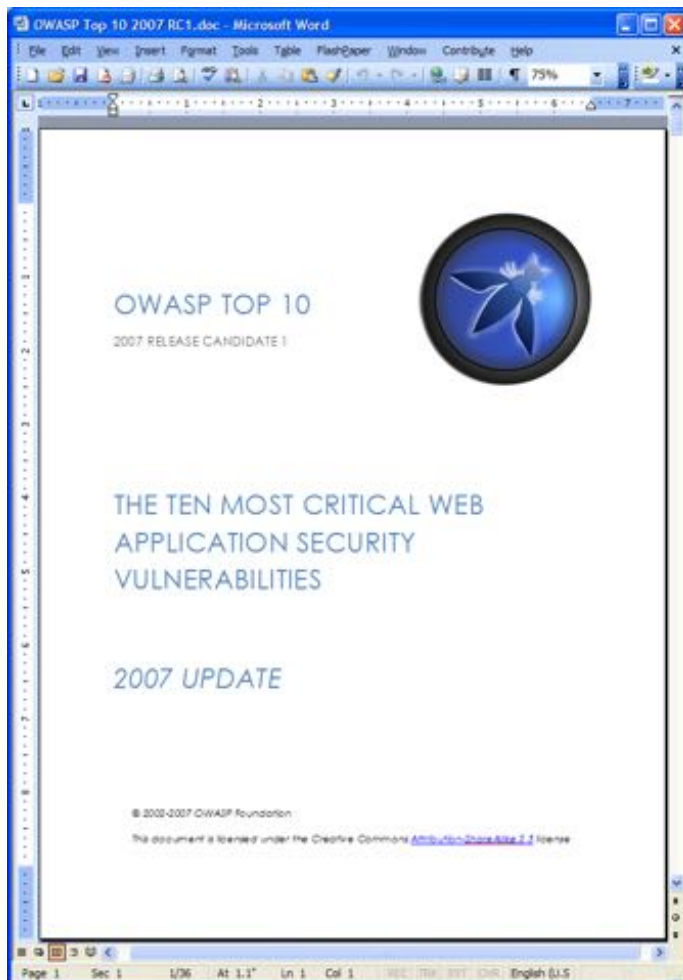
- ▶ Firewall, hardening, patching, IDS, and SSL cannot detect or stop attacks inside HTTP requests.
- ▶ Security relies on signature databases

# Agenda

- OWASP?
- OWASP Top 10, v2007RC1
- Belgium Chapter

# What is the OWASP Top 10?

- The first (but not only) things you should focus on ...



[http://www.owasp.org/index.php/Top\\_10](http://www.owasp.org/index.php/Top_10)

# OWASP Top Ten – 2007 Update

- Release Candidate 1 – Undergoing internal review
  - Will be made public by March 2007 (RC2 next week)

A1: Cross Site Scripting (XSS)

A2: Injection Flaws

A3: Malicious File Execution

A4: Insecure Direct Object Reference

A5: Cross Site Request Forgery (CSRF)

A6: Information Leakage and Improper Error Handling

A7: Broken Authentication and Session Management

A8: Insecure Cryptographic Storage

A9: Insecure Communications

A10: Failure to Restrict URL Access

RC1: [http://www.owasp.org/index.php?title=Top\\_10\\_2007](http://www.owasp.org/index.php?title=Top_10_2007)



OWASP

The Open Web Application Security Project  
<http://www.owasp.org>

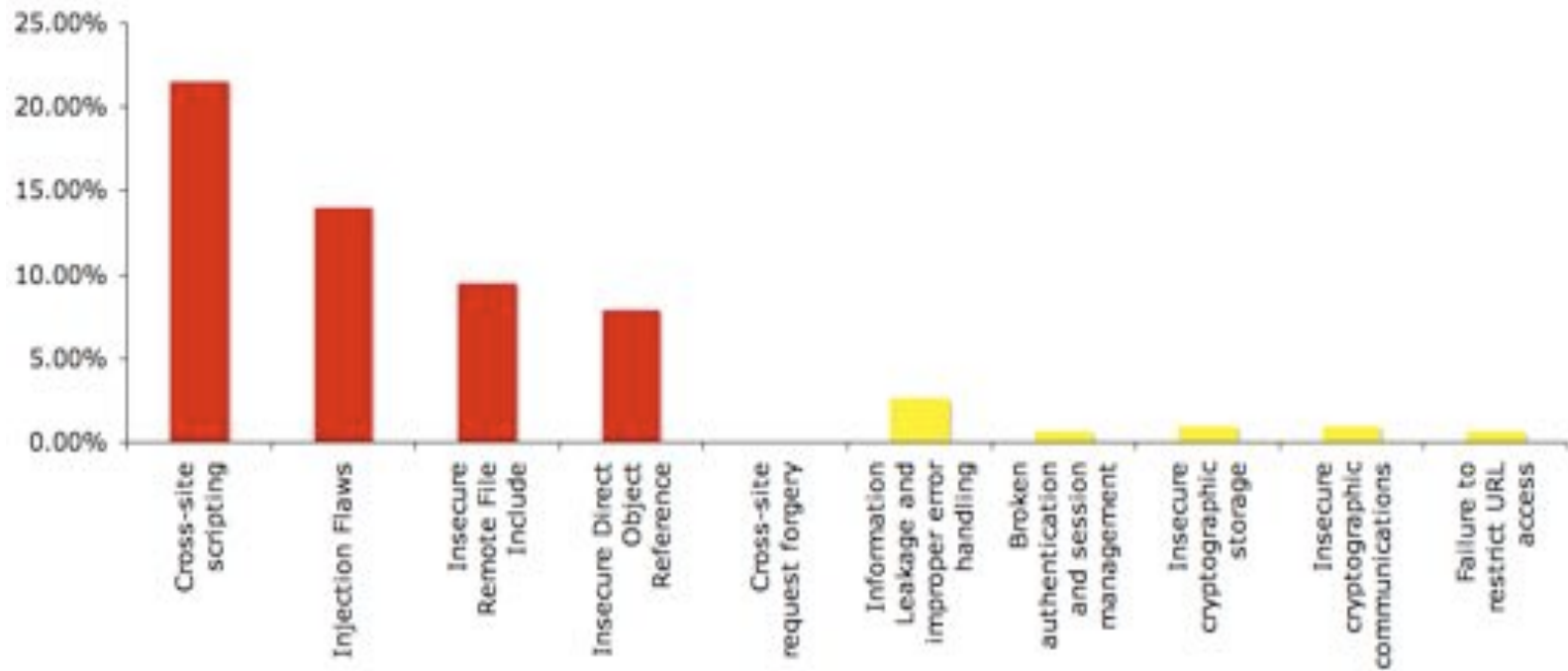
OWASP



13

# Top 10 Methodology

- Take the [MITRE Vulnerability Trends for 2006](#), and distill the Top 10 *web application security* issues



# A1. Cross-Site Scripting (XSS)

## ■ Occurs any time...

- ▶ Raw data from attacker is sent to an innocent user

## ■ Raw data...

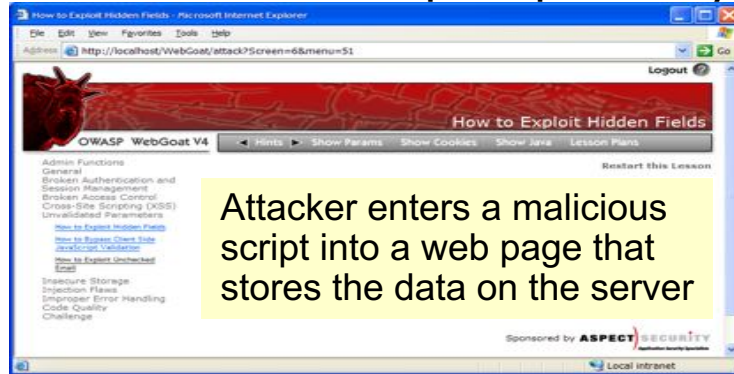
- ▶ Stored in database
- ▶ Reflected from web input (form field, hidden field, url, etc...)
- ▶ Sent directly into rich JavaScript client

## ■ Virtually every web application has this problem

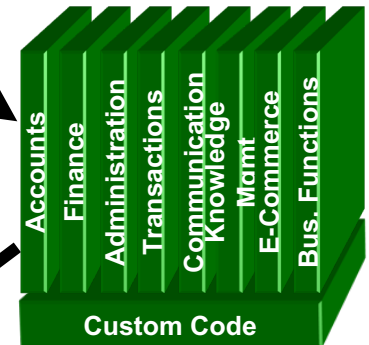
- ▶ Try this in your browser –  
`javascript:alert(document.cookie)`

# Cross-Site Scripting Illustrated

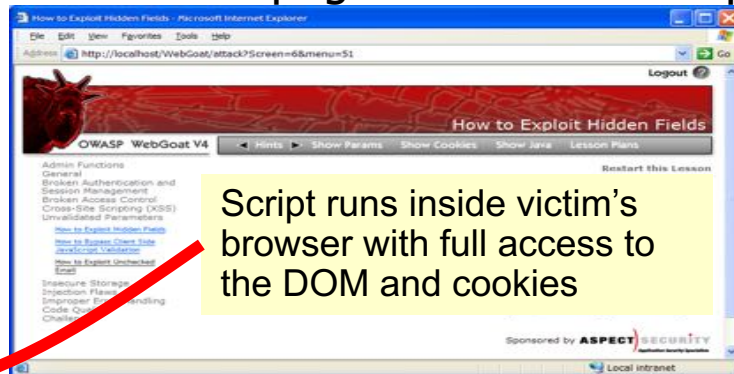
1 Attacker sets the trap – update my profile



Application with stored XSS vulnerability



2 Victim views page – sees attacker profile



3 Script silently sends attacker Victim's session cookie

## A2. Injection Flaws

### ■ Injection means...

- ▶ Tricking an application into including unintended commands in the data sent to an interpreter

### ■ Interpreters...

- ▶ Take strings and interpret them as commands
- ▶ SQL, OS Shell, LDAP, XPath, etc...

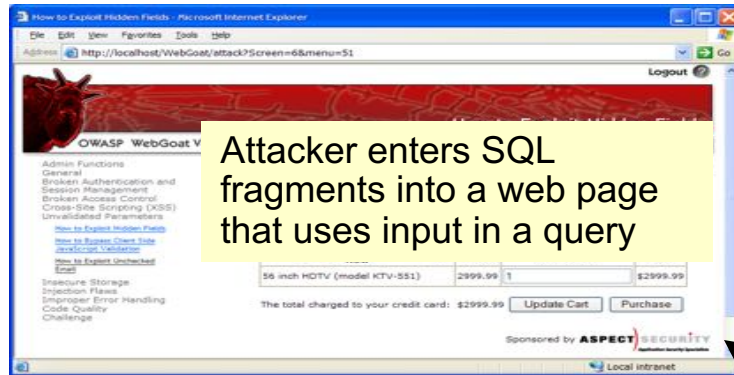
### ■ SQL injection is still quite common

- ▶ Many applications still susceptible



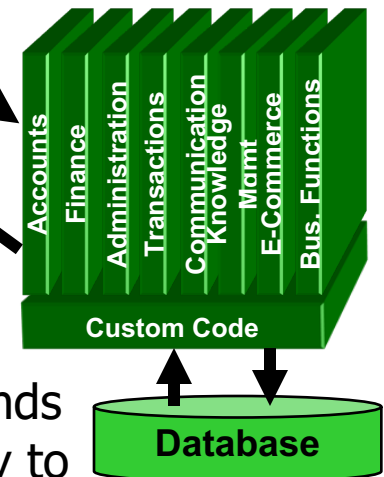
# Example: SQL Injection Illustrated

1 Attacker sends data containing SQL fragments



3 Attacker views unauthorized data

2 Application sends modified query to database, which executes it



## EXAMPLE:

```
$sql = "SELECT * FROM table WHERE id = " . $_REQUEST['id'] . """;
```

# A3: Malicious File Execution

## ■ Occurs when ...

- ▶ Attacker can influence an application to reference, upload, or create reference to a malicious file that gets executed

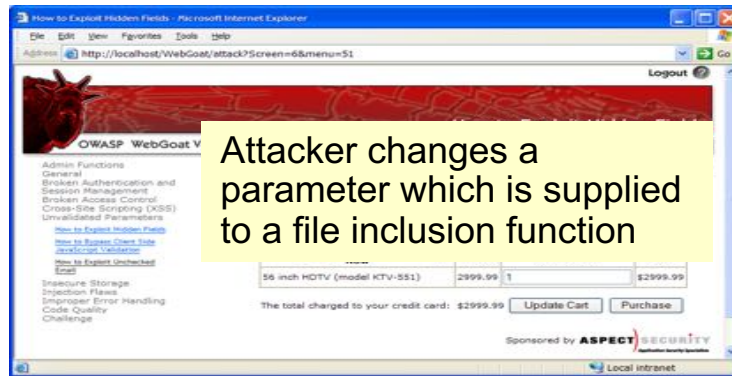
## ■ Example Scenarios

- ▶ Very frequent flaw in PHP applications where untrusted variables are used in calls like `include()`, `include_once()`, `require()`, etc.
- ▶ Application accepts name of file to execute as input, such as language choice drop down menus
- ▶ Attacker supplies unauthorized reference to code (usually an attack script)

## ■ Can occur in any framework, not just PHP: XSLT transforms, batch file includes, log files, etc.

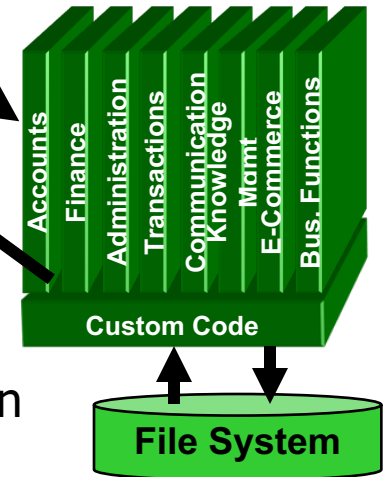
# Example: PHP Remote File Include Illustrated

- 1 Attacker sends request that specifies the path to a malicious file in a parameter



- 3 Attacker views results of executing the attack, or takes control of the affected server

- 2 PHP application includes the specified file and executes the contents



# A4. Insecure Direct Object Reference

- How do you protect access to data and other objects?
  - ▶ This is part of enforcing proper “authorization”, along with A10: Failure to Restrict URL Access
- Frequently enforced by
  - ▶ Only listing the ‘authorized’ objects for the current user
  - ▶ Hiding the object references in hidden fields
  - ▶ This is called presentation layer access control, and doesn’t work
  - ▶ Attacker simply tampers with parameter value
- For each parameter, a site needs to do 3 things
  - ▶ Verify the parameter is properly formatted
  - ▶ Verify the user is allowed to access the target object
  - ▶ Verify the requested mode of access is allowed to the target object (e.g., read, write, delete)

# Insecure Direct Object Reference Illustrated

The screenshot shows a Microsoft Internet Explorer browser window displaying an online banking account summary for a checking account (6034). The address bar contains the URL `https://www.onlinebank.com/user?acct=6065`. The page features a navigation menu, a 'Welcome' message, and a 'Year Accounts' section. The main content area displays 'Income and Expenses from Sep 26, 2004 to Jan 16, 2005' with a bar chart and a transaction table. The table lists various transactions such as 'Interest Payment', 'ATM Withdrawal', 'myBank Credit Card Bill Payment', and 'Marketing Assets Bill Payment'.

Date	Description	Category	Amount
Nov 20, 2004	Interest Payment	Interest	\$1.25
Nov 20, 2004	ATM Withdrawal, myBank, San Rafael, CA	Cash	-\$100.00
Nov 19, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	-\$100.00
Nov 16, 2004	SBC Photo Bill Payment	Phone	-\$74.23
Nov 16, 2004	myBank Credit Card Bill Payment	Credit Card	-\$2,853.57
Nov 15, 2004	ATM Withdrawal, myBank, San Rafael, CA	Cash	-\$100.00
Nov 15, 2004	myBank Payroll	Payroll	\$4,373.79
Nov 15, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	-\$100.00
Nov 4, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	-\$100.00
Nov 3, 2004	myBank Credit Card Bill Payment	Credit Card	-\$10.00
Nov 1, 2004	Marketing Assets Bill Payment	Phone	-\$13.37
Nov 1, 2004	Prudential Insurance Bill Payment	Insurance	-\$405.00
Nov 1, 2004	Chase Manhattan Mortgage Corp Bill Payment	Mortgage	-\$1,194.43
Oct 29, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	-\$100.00
Nov 16, 2004	myBank Payroll	Payroll	\$4,376.54

- Attacker notices his acct parameter is 6065  
?acct=6065
- He modifies it to a nearby number  
?acct=6066
- Attacker views the victim's account information

# A5. Cross Site Request Forgery

## ■ Cross Site Request Forgery (CSRF)

- ▶ An attack where the victim's browser is tricked into issuing a command to a vulnerable web application

## ■ Imagine...

- ▶ What if a hacker could steer your mouse and get you to click on links in your online banking application?
- ▶ What could they make you do?

## ■ Attackers can use CSRF to...

- ▶ Initiate transactions (transfer funds, logout user, close account, etc...)
- ▶ Access sensitive data
- ▶ Change account details
- ▶ And much more...

# CSRF Illustrated

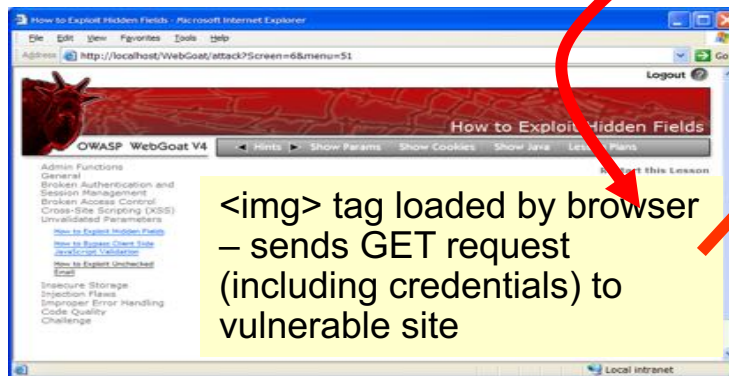
Attacker sets the trap on some website on the internet  
(or simply via an e-mail)

1

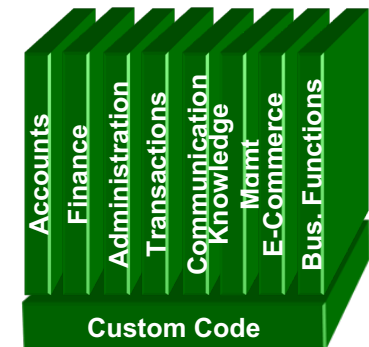


2

While logged into vulnerable site,  
victim views attacker site



Application with  
CSRF vulnerability



3

Vulnerable site sees  
legitimate request  
from victim and  
performs the action  
requested



# A6. Information Leakage and Improper Error Handling

- Web applications leak information and encounter error conditions
  - ▶ Frequently this invokes untested code paths
  - ▶ Attackers learn about your application through error messages
- Identify attacks and handle appropriately
  - ▶ Never show a user a stack trace
  - ▶ If someone is attacking you, don't keep trying to help
  - ▶ But how do you know which errors are attacks?
- Most web applications are quite fragile
  - ▶ Especially when you use a tool like WebScarab



# Improper Error Handling Illustrated

## ■ Many security mechanisms fail open

- ▶ isAuthenticated()
- ▶ isAuthorized()
- ▶ isValid()

## ■ Bad logic (i.e., fail open)

```
if (!security_test())  
    then return false  
return true
```

## ■ Good logic (i.e., fail secure)

```
if (security_test())  
    then return true  
return false
```

[Microsoft][ODBC Microsoft Access Driver] Syntax error in string in query expression 'last\_name = bob or foo'.

### Server Error in '/project' Application.

*Syntax error (missing operator) in query expression "Constantly harrassing coworkers<script>alert("DE")</script>".*

**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.

**Exception Details:** System.Data.OleDb.OleDbException: Syntax error (missing operator) in query expression "Constantly harrassing coworkers<script>alert("DE")</script>".

**Source Error:**

```
Line 131:         OleDbCommand com = new OleDbCommand(sq, con);  
Line 132:         int i = com.ExecuteNonQuery();  
Line 133:     }  
Line 134:     }  
Line 135: } catch (FormatException fe) {
```

**Source File:** c:\perf\ora\_data\depot\WebGoat\DotNet\CS\SharpDotNet\main\project\CrossSiteScripting\EdProfile.aspx.cs **Line:** 133

**Stack Trace:**

```
[OleDbException (0x80040e14): Syntax error (missing operator) in query expression "Constantly harrassing coworkers<script>alert("DE")</script>".  
System.Data.OleDb.OleDbCommand.ExecuteNonQueryForSingleResult(tagDBPARAMS dbParams, Object& executeResult)  
System.Data.OleDb.OleDbCommand.ExecuteNonQuery(Object& executeResult) -194  
System.Data.OleDb.OleDbCommand.ExecuteNonQuery(CommandBehavior behavior, Object& executeResult) +56
```

# A7. Broken Authentication and Session Mgmt

## ■ HTTP is “stateless” protocol

- ▶ Means credentials have to go with every request
- ▶ Should use SSL for everything requiring authentication

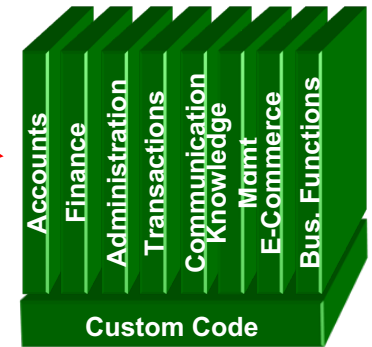
## ■ Session management

- ▶ SESSIONID used to track state since HTTP doesn't
- ▶ SESSIONID is just as good as credentials to an attacker
- ▶ Never expose SESSIONID on network, in browser, in logs, ...

## ■ Beware the side-doors

- ▶ Change my password, remember my password, forgot my password, secret question, logout, email address, etc...

# Broken Authentication Illustrated



1 User sends credentials

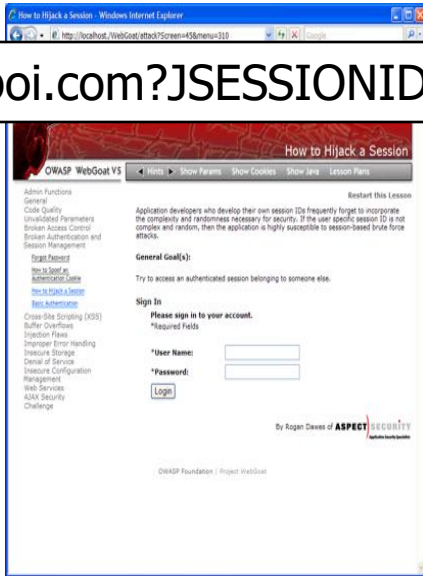
`www.boi.com?JSESSIONID=9FA1DB9EA...`

2 Site uses URL rewriting (i.e., put session in URL)

3 User clicks on a link to <http://www.hacker.com> in a forum

4 Hacker checks referer logs on [www.hacker.com](http://www.hacker.com) and finds user's JSESSIONID

5 Hacker uses JSESSIONID and takes over victim's account



# A8. Insecure Cryptographic Storage

## ■ Storing sensitive data insecurely

- ▶ Identify all sensitive data
- ▶ Identify all the places that sensitive data is stored
  - Databases, files, directories, log files, backups, etc.

## ■ Protect with appropriate mechanisms

- ▶ File encryption, database encryption, data element encryption

## ■ Use the mechanisms correctly

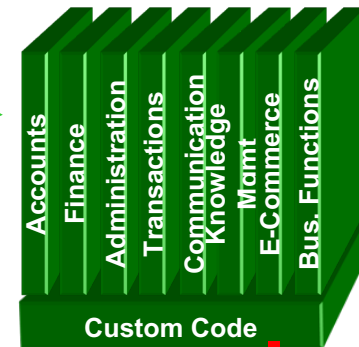
- ▶ Use standard strong algorithms
- ▶ Generate and protect keys
- ▶ Be prepared for key change

# Insecure Cryptographic Storage Illustrated



1

Victim enters credit card number in form



4

Malicious insider steals 4 million credit card numbers



Error handler logs CC details because merchant gateway is unavailable

2

Logs are accessible to all members of IT staff for debugging purposes

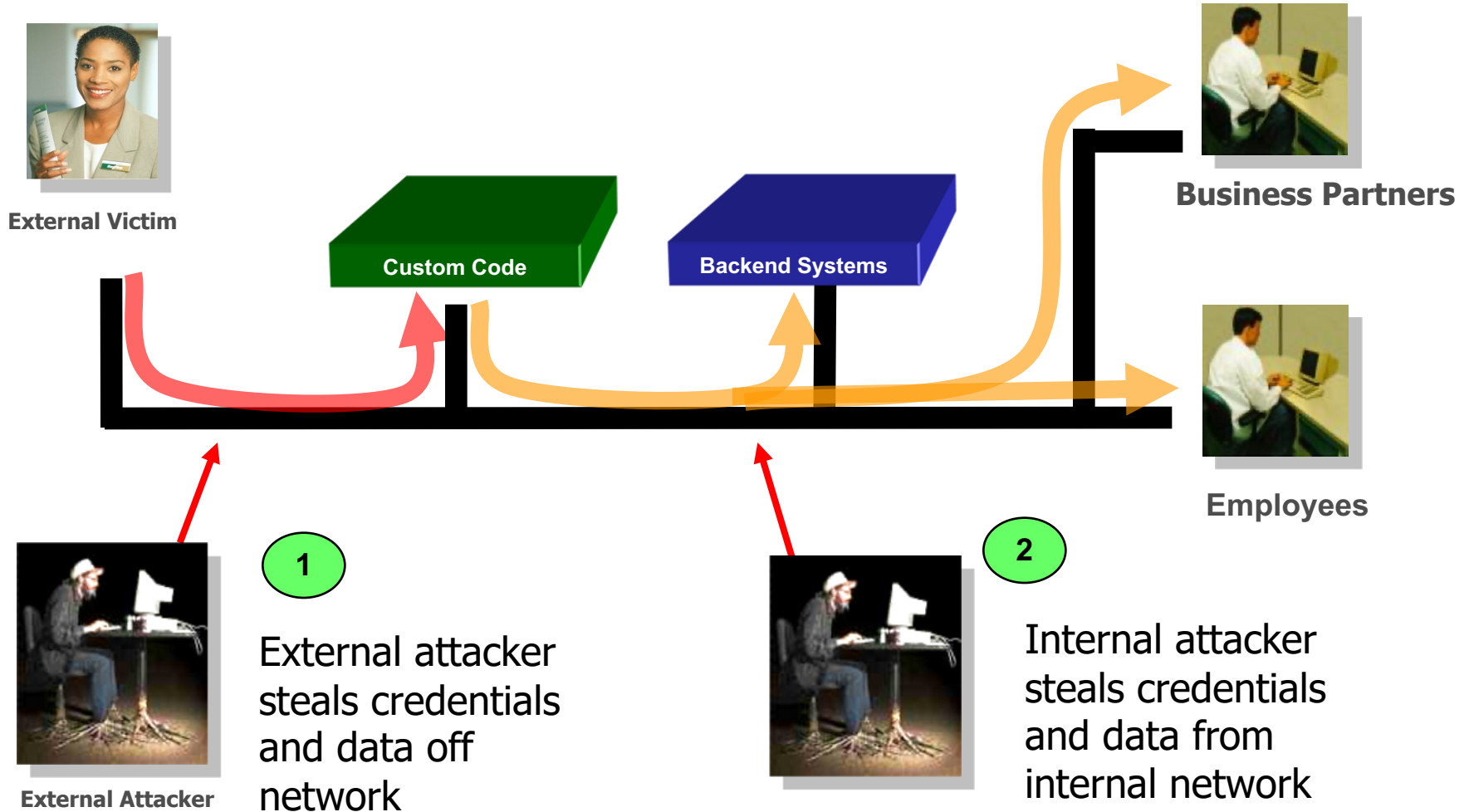
3



# A9. Insecure Communications

- Transmitting sensitive data insecurely
  - ▶ Identify all sensitive data
  - ▶ Identify all the places where sensitive data is sent
    - On the web, backend databases, business partners, internally
  
- Protect with appropriate mechanisms
  - ▶ Use SSL on all connections with sensitive data
  - ▶ Individually encrypt messages before transmission
  
- Use the mechanisms correctly
  - ▶ Use standard strong algorithms (disable old SSL alg.)
  - ▶ Manage keys/certificates properly
  - ▶ Use proven mechanisms when sufficient
    - E.g., SSL vs. XML-Encryption

# Insecure Communications Illustrated



# A10. Failure to Restrict URL Access

- How do you protect access to URLs (pages)?
  - ▶ This is part of enforcing proper “authorization”, along with A4: securing direct object references
- Frequently enforced by
  - ▶ Displaying only authorized links and menu choices
  - ▶ This is called presentation layer access control, and doesn't work
  - ▶ Attacker simply forges direct access to 'unauthorized' pages
- For each URL, a site needs to do 3 things
  - ▶ Restrict access to authenticated users (if not public)
  - ▶ Enforce any user or role based permissions (if private)
  - ▶ Completely disallow requests to unauthorized page types (e.g., config files, log files, source files, etc.)



# Failure to Restrict URL Access Illustrated

The screenshot shows a Microsoft Internet Explorer browser window displaying an online banking account summary. The address bar contains the URL `https://www.onlinebank.com/user/getAccounts`. The page content includes a sidebar with account balances for two checking accounts, a main area with a bar chart titled "Income and Expenses from Sep 26, 2004 to Jan 16, 2005", and a table of transactions. The transactions table has columns for Date, Description, Category, and Amount.

Date	Description	Category	Amount
Nov 22, 2004	Interest Payment	Interest	\$ 25
Nov 22, 2004	ATM Withdrawal, myBank, San Rafael, CA	Cash	\$100.00
Nov 29, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	\$100.00
Nov 04, 2004	TWC Phone Bill Payment	Phone	\$94.23
Nov 04, 2004	myBank Credit Card Bill Payment	Credit Card	\$2,493.01
Nov 04, 2004	ATM Withdrawal, myBank, San Rafael, CA	Cash	\$100.00
Nov 05, 2004	myBank Payroll	Payroll	\$4,373.79
Nov 05, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	\$100.00
Nov 4, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	\$100.00
Nov 3, 2004	myBank Credit Card Bill Payment	Credit Card	\$10.00
Nov 0, 2004	Working Assets Bill Payment	Phone	\$11.51
Nov 0, 2004	Preferred Insurance Bill Payment	Insurance	\$433.00
Nov 0, 2004	Chase Manhattan Mortgage Corp Bill Payment	Mortgage	\$2,184.43
Oct 28, 2004	ATM Withdrawal, myBank, San Francisco, CA	Cash	\$100.00
Nov 04, 2004	myBank Payroll	Payroll	\$4,534.94

- Attacker notices the URL indicates his role `/user/getAccounts`
- He modifies it to another directory (role) `/admin/getAccounts`, or `/manager/getAccounts`
- Attacker views more accounts than just their own

# Agenda

- OWASP?
- OWASP Top 10, v2007RC1
- Belgium Chapter

# Belgium Chapter - What do we have to offer?

- Quarterly Meetings
- Local Mailing List
- Presentations & Groups
- Open forum for discussion
- Meet fellow InfoSec professionals
- Create (Web)AppSec awareness in Belgium
- Local projects: E.g. Education Project

# OWASP near you soon:

## ■ Next BE Chapter Meeting:

▶ Tuesday May 10 2007 – (Leuven)

- Legal Aspects (Web)AppSec (Jos Dumortier – Lawfort)
- AppSec Research Topics (Lieven Desmet – KUL)

## ■ Next EU conference: OWASP EU Italy

▶ May 15th-17th - in Milan, Italy

# Stay up to date

WWW.OWASP.ORG

Belgium:

<http://www.owasp.org/index.php/Belgium>

contact: [seba@deleersnyder.eu](mailto:seba@deleersnyder.eu)

# Backup slides

# Top 10 Mapping

OWASP Top 10 2007	OWASP Top 10 2004	MITRE 2006 Raw Ranking
1. Cross Site Scripting (XSS)	4. Cross Site Scripting (XSS)	1
2. Injection Flaws	6. Injection Flaws	2
3. Insecure Remote File Include (NEW)		3
4. Insecure Direct Object Reference	2. Broken Access Control (split in 2007 T10)	5
5. Cross Site Request Forgery (CSRF) (NEW)		36
6. Info Leakage and Improper Error Handling	7. Improper Error Handling	6
7. Broken Auth. and Session Management	3. Broken Authentication and Session Management	14
8. Insecure Cryptographic Storage	8. Insecure Storage	8
9. Insecure Communications (NEW)	Discussed under 10	8
10. Failure to Restrict URL Access	2. Broken Access Control (split in 2007 T10)	14
	1. Unvalidated Input	7
	5. Buffer Overflows	4, 8, and 10
	9. Denial of Service	17
	10. Insecure Configuration Management	29