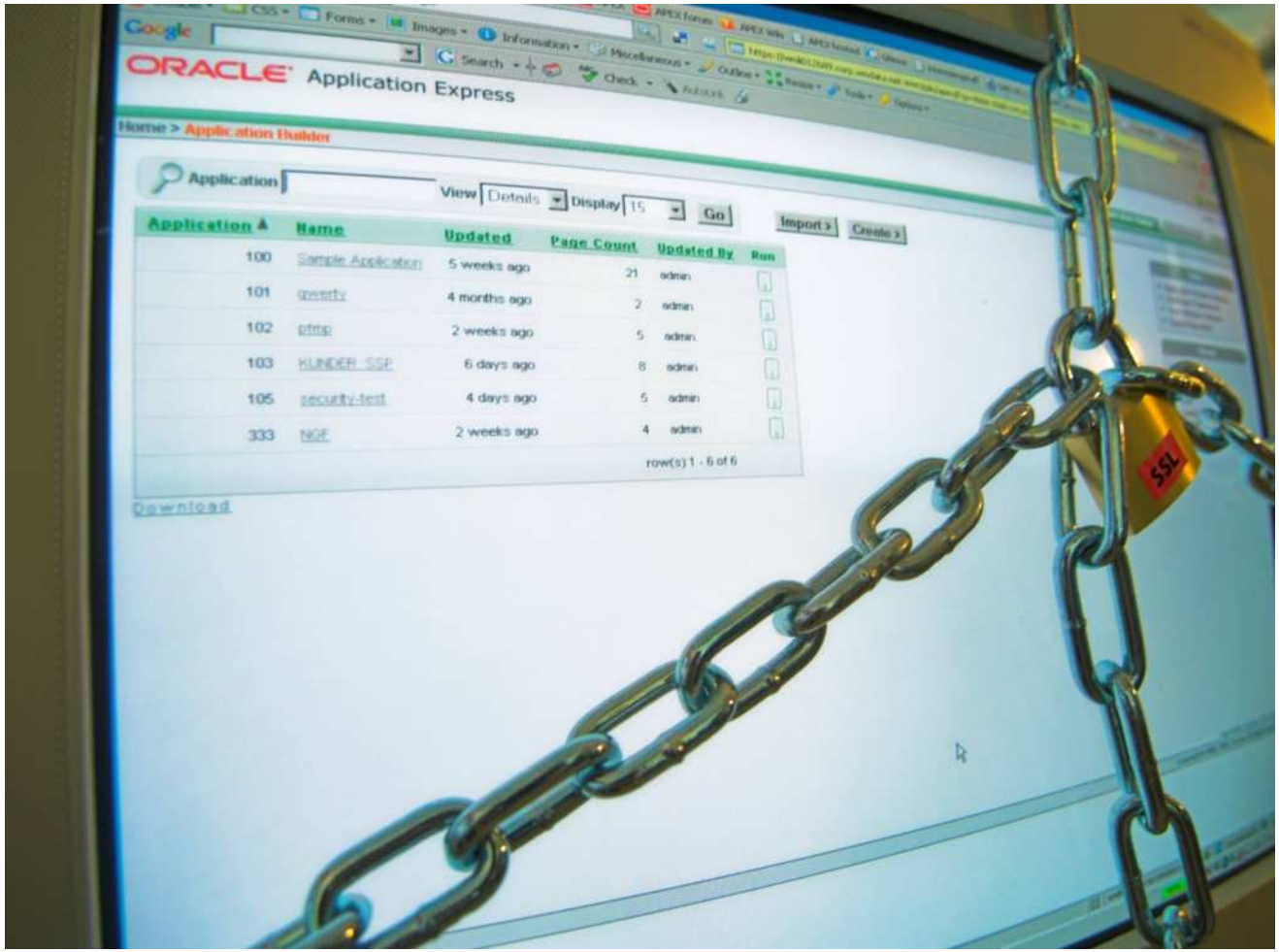
How to Make Your Oracle APEX Application Secure



Peter Lorenzen

Technology Manager

WM-data Denmark

a LogicaCMG Company

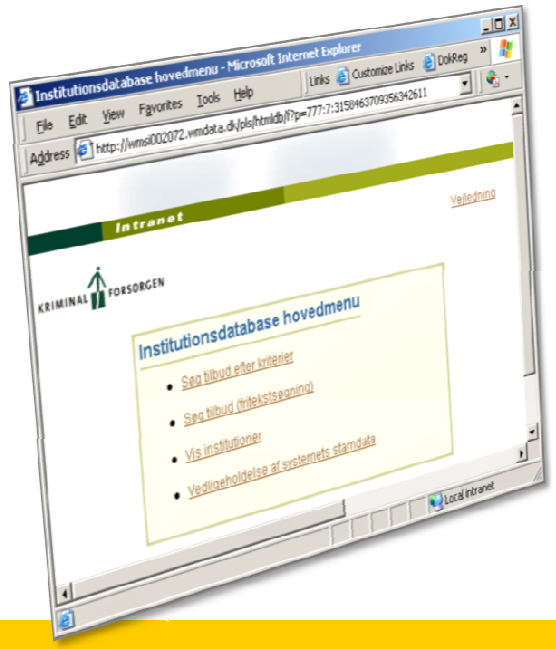
peloz@wmdata.com

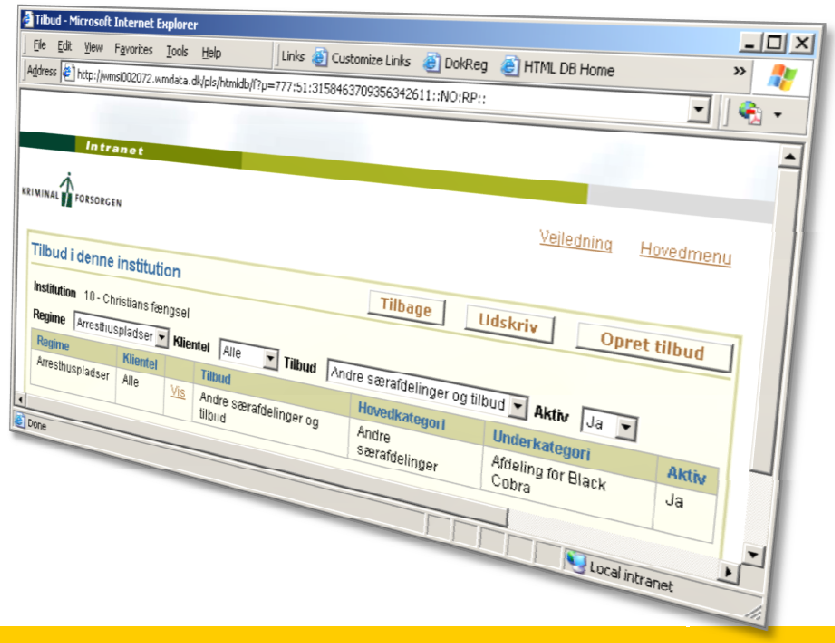
Presentation

- Target audience is **developers**
- Focus is on how to prevent **hackers** from gaining access
- In terms of what I believe an **APEX developer in a small shop**, without a fulltime security expert or DBA, should know
- More an **overview of security threats and countermeasures** than a thorough analysis
- **Point you to resources** with more information about the different subjects
- Assumption: An application that
  - is **accessed from the Internet**
  - contains **valuable and secret information**

APEX Project References

- The Danish Department of Prisons and Probation uses APEX in the process of deciding in which facility a client should serve
- RTX Telecom uses APEX to control DECT cordless telephones in Rumania
- Naturgas Fyn is a provider of natural gas in Denmark. We have developed a system that calculates the amount of gas that is needed from each gas provider the following day



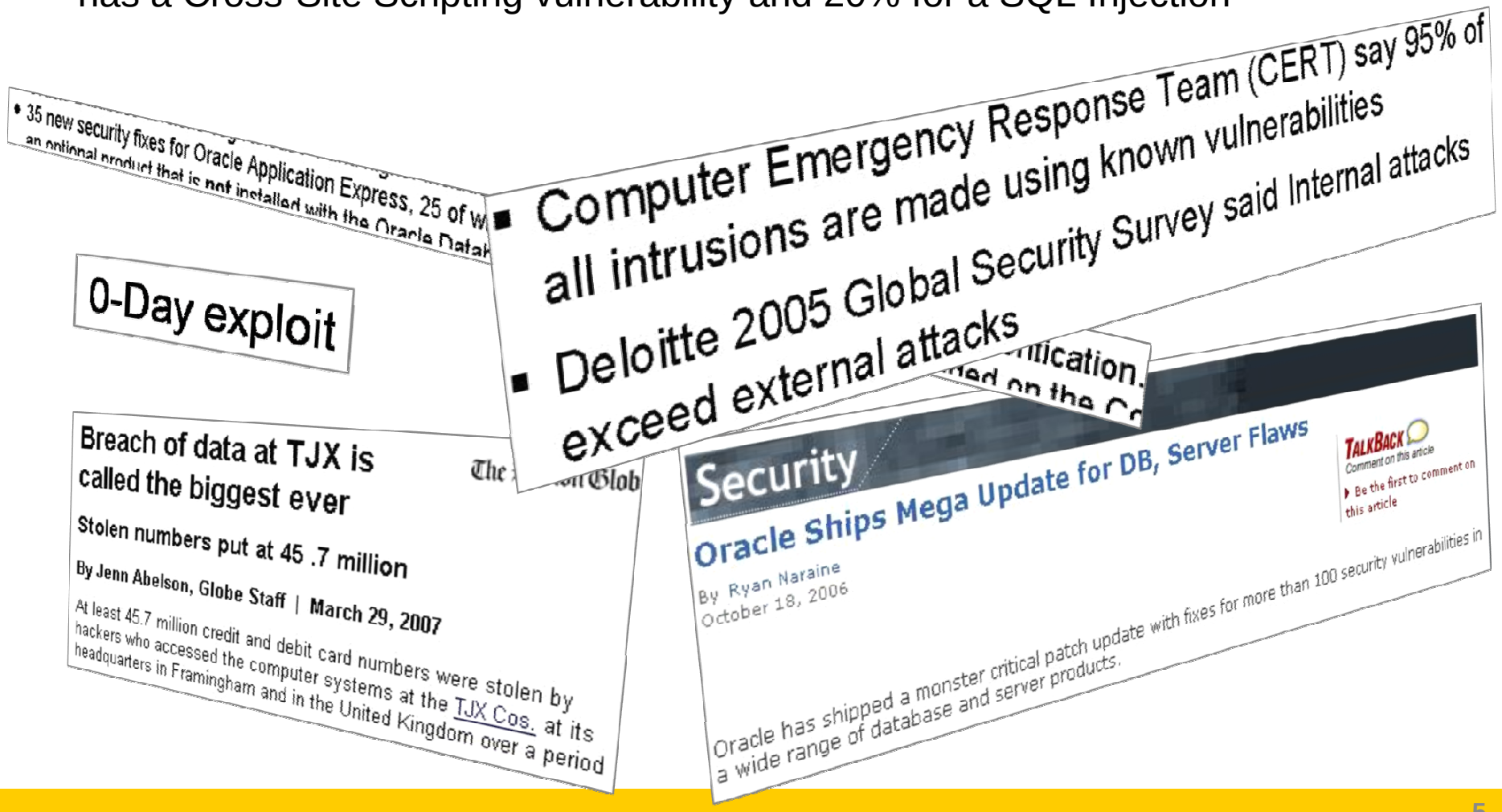


Agenda

- **Intro**
- **Architecture**
  - HTTP Servers
  - Choosing an Architecture
- **Hardening the Architecture**
  - Patching
  - Hardening the Database
  - Hardening the HTTP Web Server
- **Specific Threats**
  - Cross-Site Scripting
  - SQL Injection
- **Hardening APEX**
  - Miscellaneous
- **Conclusion**

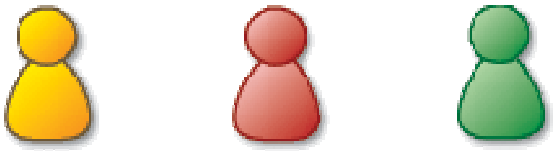
Intro – Security, what security?

A security company estimates that there are a 71% likelihood that a Website has a Cross-Site Scripting vulnerability and 20% for a SQL Injection



Intro

- Think about security from the beginning of a project
- Plan security – Architecture etc.
- Make sure people knows the security basic
- Have people that is responsible for security, patching etc.



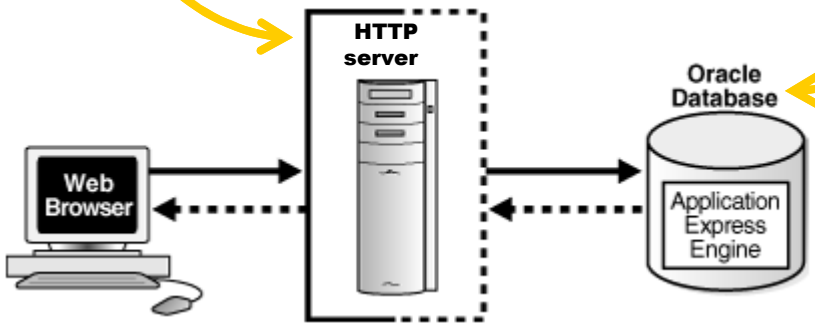
APEX is secure, developers makes it insecure ☺

Architecture

APEX Components

- Oracle HTTP Server (Database Companion CD)
- Oracle HTTP Server (Oracle Application Server)
- Oracle XML DB HTTP Server

- Oracle 9i/10g/11g Database
- ~~Oracle Express Edition~~



There is such a thing as too cheap

Architecture

Which HTTP Server to Use?

	Oracle HTTP Server (OHS) ✓	Oracle XML DB HTTP Server
Technology	Apache 1.3.x	Developed by Oracle. Builds on the Oracle Shared Server architecture
Database “connection”	mod_plsql	Embedded PL/SQL Gateway

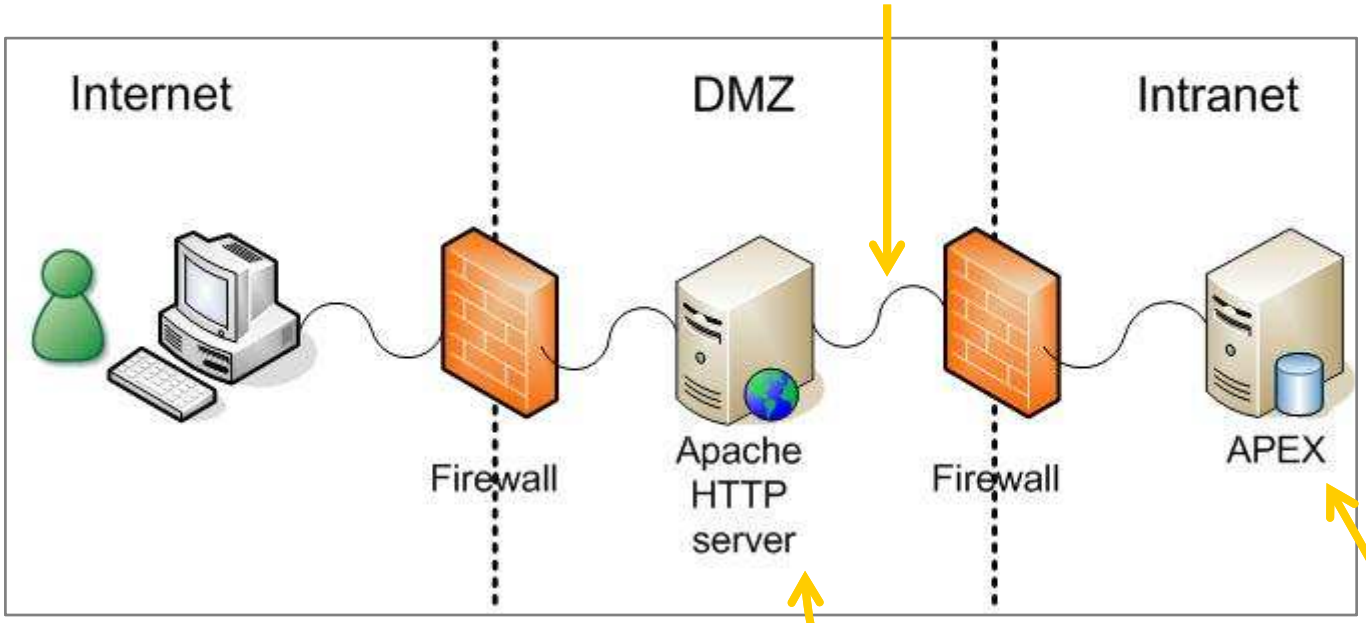
Use **known** and **proven** technology

Architecture

"Security is an architecture, not an appliance" - Art Wittman

Minimum

Only HTTP communication



Proxy HTTP Server

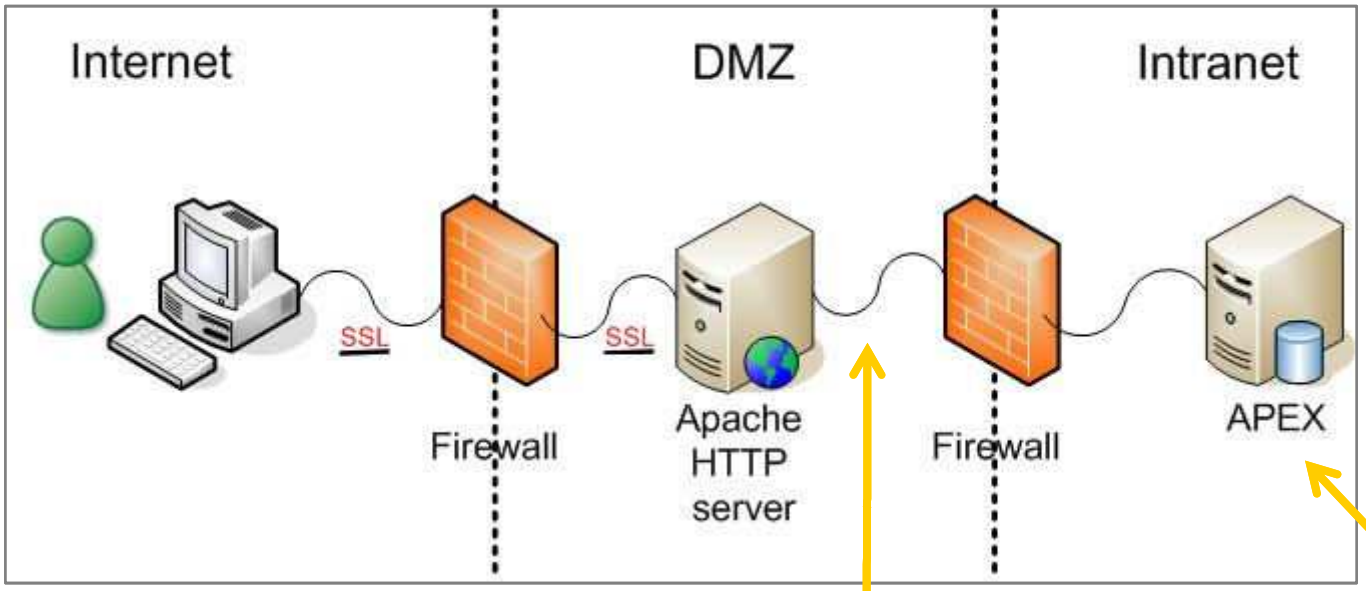
- Standard Apache 1.3/2.0 HTTP Server
- OHS based on an Apache 2.0.x HTTP Server

mod_proxy

Database + HTTP server

Architecture

Using Secure Sockets Layer (SSL) encryption



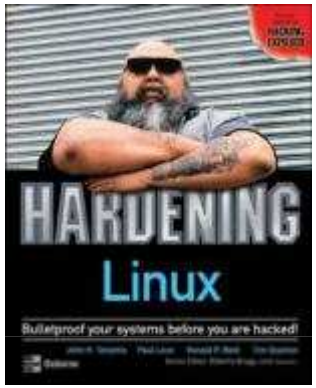
SSL?

Database +
HTTP server

Security measures should **match** the risk and the value of the secured application/data

Hardening the Architecture

- Patch, Patch, Patch
  - Critical Patch Update (CPU)
  - Oracle Security Alerts
  - Remember regular Patch Sets
  - The Oracle HTTP Server – Patches from Oracle
  - Standard Apache HTTP Servers – Patches from Apache
  - Remember the OS
  - Patching can be difficult!

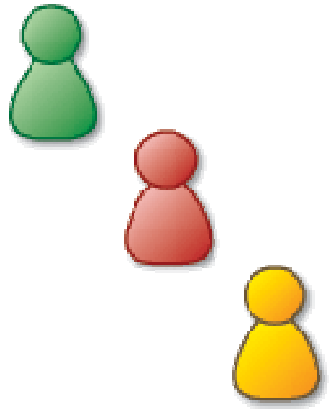


Patching should be part of the **daily operations**

Hardening the Architecture

- Hardening the Database
  - Do **not** use the free Express Edition (XE) database
- The simple stuff
  - Follow the principle of least privilege
  - Lock or remove unused users
  - Use sensible passwords
  - SYS password != SYSTEM password
- Must-reads
  - Oracle Database Security Checklist
  - “Hacking and Securing Oracle - A Guide To Oracle Security”

    by Pete Finnigan
- A good place to start
  - Oracles Project Lockdown



Use **checklists** and adopt **best practices**

Hardening the Architecture

- Hardening the Apache HTTP Web Server
  - Remove pre-loaded modules
  - Remove pre-installed content
  - Don't publicize names/versions of your running software

ServerSignature Off (Removes server information from error pages)

ServerTokens Prod (Removes server version from the HTTP header)

- Comprehensive Checklists
  - "Securing Oracle Application Server"

    by Caleb Sima
  - "Hardening Oracle Application Server 9i and 10g"

    by Alexander Kornbrust

Give away as **little** as possible about yourself

Specific Threats - Cross-Site Scripting (XSS)

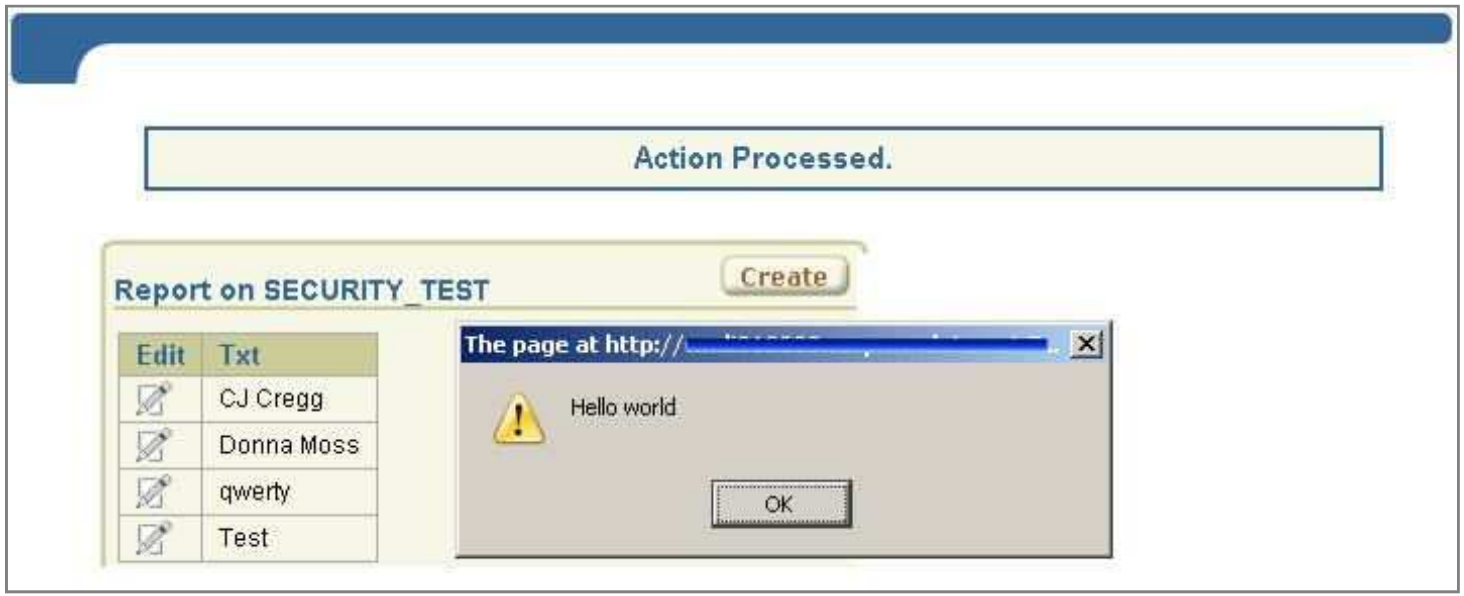
- Simple definition
  - Attacker injects JavaScript in an application in order to attack **other users**
  - Ex. Stealing data, Hijacking session token, Performing unauthorized actions
- Many types of XSS
  - Stored XSS (JavaScript in database)
  - Reflected XSS (Embedded JavaScript in URL request)
  - Stored XSS in uploaded files (HTML, Text file with .jpg extension, etc.)
- XSS is often not that dangerous on it's own, but combined with bugs in a browser, a virus or a worm it can be serious

Specific Threats - Cross-Site Scripting

- Quick example in APEX
  - Create a Form on a table of type “Form on a Table with Report”
  - Run the Report and create a row with this data in a VARCHAR2 column

Test<script>alert(‘Hello world’);</script>

  - When you press Create and branch back to the Report the JavaScript is run

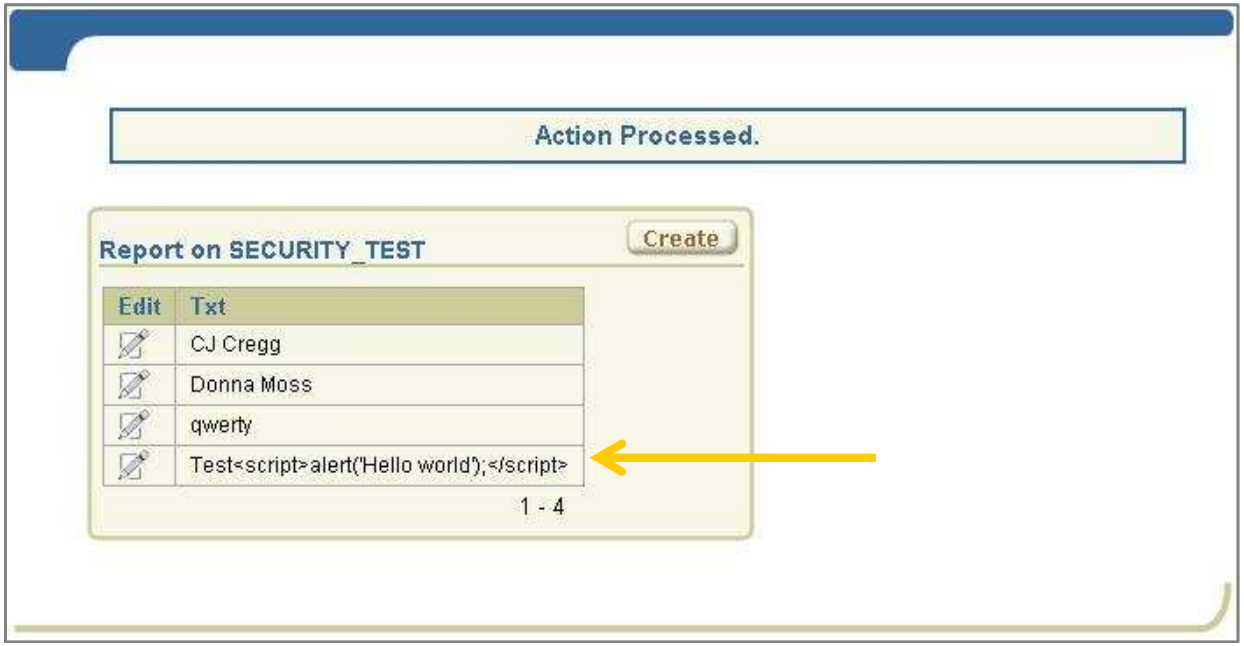


Specific Threats - Cross-Site Scripting

- Fix: Escape Special characters like <,>,&
- Change Display as

Standard Report Column

Display as text (escape special characters, does not save state)



Specific Threats - Cross-Site Scripting

- **Escaping is the weapon** of choice when dealing with XSS threats
- Escape all output
- The page source will now look like this

```
Test&lt;script&gt;alert('Hello world');&lt;/script&gt;
```

- In PL/SQL use this function: HTF.escape_sc
- Read about safe items in the User's Guide

Don't trust **any** input from the end-user

Specific Threats - SQL Injection

- Definition
  - An attacker inputs extra SQL in an application
- Simple example in APEX
  - Report based on a SQL Query

```
select  job, sal from emp where ename = '&P1_ENAME.'
```

  - The P1_ENAME item is input by a user
  - If an user input the text below **all** rows will be shown

```
qwerty' or 1=1--
```

  - The fix for this specific situation is to **use bind variables**

```
select  job, sal from emp where ename = :P1_ENAME
```

Specific Threats - SQL Injection

- Take care when an **end-user** can input text that is used in DML
- Watch out for **concatenation** of user input in DML
- Take care when using **Dynamic SQL**

  DBMS_SQL
  or
  Native Dynamic SQL e.g. Execute Immediate

- Validate end-user input:
  - Check for max. length
  - Check for parentheses, comments (--, /* */)
  - Validate the input against a table

Always use Bind Variables!

Hardening APEX

- Session State Protection (SSP)
- APEX URL

42

f?p=101:7:2564092426426::::**P7_USER_ID:99**

- APEX URL with SSP checksum

f?p=101:7:2564092426426::::**P7_USER_ID:99**&cs=38D6164631F9754257F3

- Use APEX_UTIL.prepare_url to generate checksum from PL/SQL
- SSP should **not be the only** security measure!
  - Also check in the database
    - Via triggers
    - Via a view layer and instead of triggers
    - Virtual Private Database (VPD)

Always use Session State Protection

Hardening APEX

- Security Options in the Administration Services

  (Options for you production system)

  – Disable Administrator Login

  – Disable Workspace Login

  – Restrict Access by IP Address

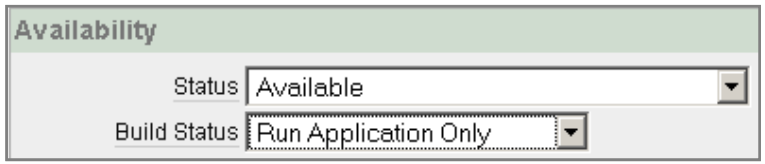
  – Workspace Password Policy

- Version 3.1 will contain a **Runtime Installation** that probably will alleviate most of this

- Miscellaneous

  – Debugging should be disabled

  – Build Status should be Run Application Only



Lock down your production system

Hardening APEX

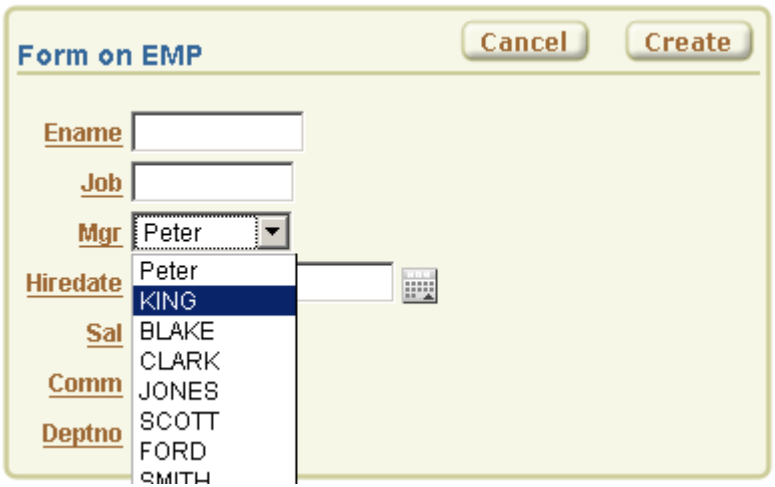
- **Obfuscate** the APEX_PUBLIC_USER Password
  - Use the dadTool.pl script located in ORACLE_HOME\Apache\modplsql\conf
  - If you use marvel.conf **rename it temporarily** to dads.conf

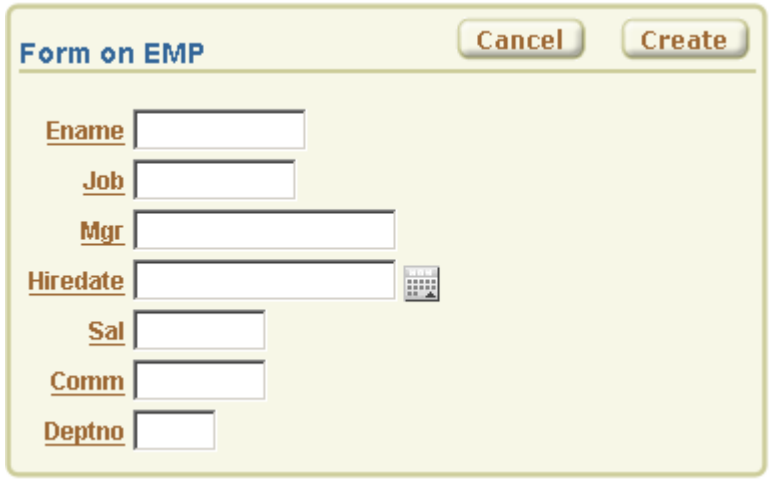
```
<Location /pls/apex>
 Order deny,allow
 PlsqlDocumentPath docs
 AllowOverride None
 PlsqlDocumentProcedure       wwv_flow_file_mgr.process_download
 PlsqlDatabaseConnectString   wmli012689.corp.wmdata.net:1521:orcl
 PlsqlNLSLanguage             AMERICAN_AMERICA.AL32UTF8
 PlsqlAuthenticationMode      Basic
 SetHandler                   pls_handler
 PlsqlDocumentTablename       wwv_flow_file_objects$
 PlsqlDatabaseUsername        APEX_PUBLIC_USER
 PlsqlDefaultPage             apex
 PlsqlDatabasePassword        oracle1
 Allow from all
</Location>
```

Miscellaneous

- Don’t trust **any** input from the end-user
  - All JavaScript and HTML can be **changed**
  - For examples try the **Web Developer** or the **Firebug** Firefox add-on
  - Do **all** validations in the database

Example using the Web Developer Firefox add-on





Secure Sockets Layer (SSL) encryption

- Check How-to's on the APEX Wiki
  - Using SSL with the Oracle HTTP Server
  - Using SSL with the Oracle XML DB HTTP Server

Conclusion

- Security is important
- Don’t think of security only in APEX but for your whole architecture
- Create a sensible architecture
- Use SSL encryption
- Patch everything
- Harden the database and the Apache HTTP Server
- Escape output to prevent Cross-Site Scripting
- Validate input to prevent SQL Injection etc.
- Use Session State Protection
- Lock down your production system
- Obfuscate the APEX_PUBLIC_USER password
- Don’t trust JavaScript validations, hidden items, check boxes, etc.

Questions?

Contact Information

Peter Lorenzen
peloz@wmdata.com

For More Information

- **CPU and Security Alerts**
  http://tinyurl.com/5dhto
- **Oracle Database Security Checklist**
  http://tinyurl.com/ytake2
- **“Hacking and Securing Oracle - A Guide To Oracle Security” by Pete Finnigan**
  http://tinyurl.com/28jrt7
- **Oracles Project Lockdown**
  http://tinyurl.com/24s4nf
- **“Securing Oracle Application Server” by Caleb Sima**
  http://tinyurl.com/2ey89a
- **“Hardening Oracle Application Server 9i and 10g” by Alexander Kornbrust**
  http://tinyurl.com/2x5h3h
- **APEX Wiki**
  http://tinyurl.com/2zosrp