For those who haven't met Dave or myself, we have been working in various sectors of IT for a number of years, recently we've been part of the IT Security team within Bankwest. Some of the activities we perform in this role include:
- Technical online crime prevention and response
- Information security due diligence
- Digital forensics
- Security Education
- and Security Assessments, including vulnerability assessments and application security assessments.

There are a number of reasons why performing application security assessments against your own systems is important. We consider it part of a risk-based approach to the construction of applications, and under certain regulatory bodies are required to test certain systems, such as the Internet Banking System.
Other regulatory requirements that may be related to your own apps include ensuring appropriate steps have been taken to protect personal or sensitive information under the Privacy act. If you process credit card information you may have to comply with the payment card industry's data security standard too.

Apart from highlighting important testing tips and tricks, today's session is more about demonstrating real-world impacts of these issues. We often have the discussions at work: "Well, if we know there's a vulnerability, why would we have to demonstrate it to anyone?" and unfortunately sometimes developers are unclear of the impact of discovered vulnerabilities, or when trying to gain budget to fix vulnerabilities within your systems you need to demonstrate what the vulnerability is actually capable of doing.

Sometimes after reporting on these vulnerabilities for so long people just glaze over when they see the findings.. Sometimes it's useful to give them a refresher.

To truly demonstrate some of the "very bad things" we're going to explain and demonstrate how cross-site scripting can lead to allowing an attacker to jump on your session.
Then we'll give a brief introduction to when you combine cross-site scripting with BeEF.
Then demonstrating what happens when you start to utilise exploitation tools, such as Metasploit, with web vulnerabilities.
Finally we'll show some useful SQL injection attacks, not just the usual ' or 1=1, but utilising some Blind SQL injection techniques to gather info out of the underlying SQL database.

Before getting into the nitty gritty we thought we should give a quick history on a few core concepts we'll be exploring today.
- cross site scripting and
- SQL injection.

There are plenty of examples for injection attacks, some pretty interesting examples from the last couple of years.
The ASPROX botnet that leveraged SQL injection to display malicious javascript on websites hit numerous websites over the past couple of years, this included Adobe, Greaterunion and Sony.
One of the biggest SQL injection attacks, as far as breached records goes, dates back to 2004 with the CardSystems breach. At least 263,000 CC numbers stolen and 40 million more exposed.
http://www.xiom.com/whid-2004-17

The demos today will be based around a fictional app that DT was kind enough to develop. As a budding developer you might be able to excuse him for his lax security skills, perhaps.
The SCOATS app itself is a typical web app, unauthenticated users can't see any content, so the first step is to investigate the login page.
So let's look at the app.

WHID: the 3rd top attack method, after SQL Injection and "Unknown".

Combine a fairly trivial vulnerability with a sprinkling of social engineering….

And what do you get? :O []

Now this method of exploiting XSS is fairly old-school. So now we'll show some more modern methods of exploiting cross-site scripting vulnerabilities…

Let me introduce you to BeEF, or the [] Browser [] Exploitation [] Framework.
BeEF is a tool used by penetration testers or system administrators to investigate additional attack vectors when assessing the posture of a target. The framework is actually used to target web-site users as opposed to websites itself.
*Tangentially, Did you know that looking at meat makes you calm?

If you're in the business of infecting multiple users, or tracking multiple sessions, tools like BeEF make this a lot easier to handle. []
The power comes from the modules, and in this case, modules can be loaded and used to target all sorts of other vulnerabilities, effectively turning BeEF hooked browsers into beachheads for launching of other commands into other areas of the network.[]
BeEF is a powerful tool to demonstrate how a trivial vulnerability, such as cross-site scripting, can explode into something larger.[]

Such as…

BeEF is a great tool to demonstrate what can be achieved within the sphere of the browser. But often security testers (or attackers) want to go deeper than just the browser.
This is where penetration testing frameworks come into play, such as CORE impact, Immunity's Canvas, and Metasploit.

Metasploit comes in a number of flavours, from the free/open source derivative which we will be demonstrating today..

through to their PRO version, which includes web server scanning & exploitation functionality, reporting, workflows, and client-side social engineering components, to name some of their features.

The two primary concepts of Metasploit are the exploit, which is code that will penetrate a system via a vulnerability ….

And the payload which is delivered after the vulnerability has been exploited.

A simple example, and what we'll demonstrate.

This is probably why drive-by-downloads are the largest point of injection for malware distribution.

Now we'll move onto Blind SQL Injection.
***Blindfolded Typing Competition**
circa 1940, Paris, France - Blindfolded Typing Competition

Demonstrate injection in login page. Both cases.

I'm thinking of a number between 1 and 100…
(65?)
Get the audience to guess – answer with "lower" or "higher".

An efficient way to search an ordered list is the binary search.
"Guess" a value half way through the list.
Compare the guessed value to what you are searching for.
If the value being searched for is greater than the guessed value, then search space is now the top half of the original list.
If the value being searched for is less than the guessed value, then the search space is not the bottom half of the original list.
Shake and repeat.

This is probably why drive-by-downloads are the largest point of injection for malware distribution.

Okay, so XSS is bad and it should be fixed.
But how?
http://www.flickr.com/photos/ethel theaardvark/5536233301/

Start by thinking about security within your software development lifecycle, have a look at Microsoft Security Development Lifecycle or..

OWASPs Comprehensive, Lightweight Application Security Process, or CLASP.

If you're doing a lot of serious development don't forget to check out the OWASP Enterprise Security API project, or ESAPI.
ESAPI IS NOT a framework (like Spring or Struts)
ESAPI IS a set of foundational security controls
To allow for language-specific differences, ESAPI is built on particular design principles:
- ESAPI defines a set of security control interfaces
- There is a reference implementation for each of these controls (a default, simple implementation)
- There are your own optional implementation for these controls. (enterprise federated authentication?)

Whilst a bit aged, the OWASP Development Guide is still a great resource.
Latest stable release was 2005. They are currently working on a new version.

Test your apps, if you can't get pen testers, allow some of your developers to review the OWASP TEsting Guide

Don't know where to start? Check out OWASP's Software Assurance Maturity Model.
OWASP's SAMM is a great place to help you understand what your software security processes and maturity look like right now, and helps construct a roadmap to improve your stance over time

This is probably why drive-by-downloads are the largest point of injection for malware distribution.

Questions?