The ABC’s website has been hacked … but how?

Yesterday the ABC was attacked, supposedly in response to the network giving coverage to an Australian visit by the controversial Dutch politician Geert Wilders.

A Twitter user, “Phr0zenMyst”, claimed responsibility for the hack, which exposed the personal information - including usernames, locations and email addresses - of nearly 50,000 people.

Needless to say, there are a few questions to be answered about this episode.

High and low risk …

Why should we be concerned that some passwords and user IDs from an obscure ABC website (the site, relating to the ABC TV show Making Australia Happy, has now been shut down) were made public?

The main reason is that passwords tend to be re-used, given we have so many systems we need to use passwords for.

If re-use is carried out in a disciplined manner it can be a workable way of minimising the number of passwords we have to remember: we might re-use the same password for low risk situations – such as for commenting on an ABC website – but have a different password for each of our email and banking accounts.

But many of us re-use the same password for low- and high-risk accounts; which means compromising a relatively low-risk website may have consequences for higher risk websites.

SQL injection

We don’t know the details yet as to how the data was extracted, but recent similar exploits have used what is known as an SQL injection attack, and there’s a reasonable chance that was what was used here.

Most web systems, as well as serving web pages, also use a database system, which is distinct from the web server and will often be on a separate physical device. During ordinary operation, data is extracted or written to the database via commands on the web server using a language called the Structured Query Language (SQL).

So, when we enter a user name and password a query is made of the database via commands embedded within the webpage using SQL. In an SQL injection attack, the webpage is bypassed and the attacker sends SQL queries directly to the database.

In an unprotected system this is surprisingly easy to do and this may well have been how the data was extracted.

But we don’t yet know for sure that this occurred in the ABC attack, and it may well be something else. But either way the database appears to have been inadequately protected and contributor data, including user IDs and passwords, were leaked.

Making a hash

We know a lot more about the other security weakness: how the passwords were stored.

In any system that uses passwords, other than the most trivially insecure, the password is never stored as plain text. It goes through a cryptographic function known as a hash.
A hash takes an arbitrarily long sequence (in this case the password) and generates a fixed length string of bits, known as the “hash”. The same hash is always returned for the same input.

In the simplest case, the user enters a password, the web server calculates the hash of the password, the stored hash is retrieved from the database, the two are compared and, if equal, the user has been verified.

Hashes work on the assumption that knowing the hash of a value tells you nothing about the original value.

Unfortunately, that assumption is no longer valid. There are plenty of readily available systems – such as L0phtCrack and John the Ripper – that, given a hash value, will rapidly return the corresponding input for simple passwords such as those that use only lower-case alphabetic characters. To make passwords more secure, they should be “salted”.

“Salting” is the process of adding a large random string of characters to a password before it is hashed. When the web server stores or seeks to verify the password, it adds the “salt” to the input password and then hashes it.

Salting makes extracting a password knowing only its hash much more challenging. From reports by people who have started extracting passwords from the posted files, it appears salting was not used in the ABC website’s password protection.

**Today’s lesson is …**

So that’s roughly what we know or can speculate as to how yesterday’s hack was done. Finally, what lessons are there to be learnt in this episode?

There are lessons for both users and developers.

Users need to be very careful about re-using passwords. A password used for a relatively low-risk site should never be re-used for sensitive sites such as banking or emails.

Users should try and get into the habit of generating good, but easy-to-remember passwords. The great cartoon below (by xkcd) gives an example of how, and there are plenty of other techniques as well.
Developers too need to assume that any system they attach to the internet will be probed for weaknesses that, if found, will be exploited. Security should be an integral part of the design.

That design should make sure that a database attached to a system is secured properly, while ensuring sensitive data such as passwords is salted before hashing.

There’s nothing new with these lessons. These things have been known for years. With all that is now known about security, it’s worrying that we still see these kinds of exploits.