



"Try Before You Buy"

A brochure about your new car will provide you with specifications, pictures of the inside and outside, colour options, equipment list and possibly some award references. The website will offer you a 360° view from the driver's seat and if you like, you can change the colours and alloy wheels to suit your taste. Then of course similar information is available about similar models from other manufacturers. While this information is useful in at least narrowing down the field, I have yet to meet someone that has picked up the phone and purchased a car solely based on brochureware.

The missing and most important element in the purchasing process is the test drive. You need to physically sit in the seat, turn buttons, switches and leavers, see how much head room you have, how big the luggage space is and most importantly, how the car feels and handles on the road. An important aspect of the test is the need to drive the same model as the one you're intending to buy. You wouldn't drive a 4 cylinders automatic model when you're interested in the V6 manual with turbo. Why test the two wheel drive version when you want to feel the handling of the all wheel drive model.

This experience carries significant weight in assessing the car's value for money proposition and collectively leads to the emotional or academic decision to buy.

In the shrink-wrapped software industry, software developers are seeing great benefit in offering trial or demonstration versions of their product. Here the prospect is that by allowing a potential customer to try the software before committing to buy, the likelihood of a sale is vastly higher. Like in the car analogy, a brochure with specifications, screen images and a feature list provides limited input to the decision to purchase the software package. A prospective buyer will gain far greater insight about the software and suitability for their needs if given the opportunity to use it in their environment and in the context of their specific business or need.

Considerations such as navigation within the software, the way data is entered and information is retrieved, the ease of use, the depth and spectrum of reference material provided, access to Help features and other aspects all significantly contribute to the experience and ultimately, the decision to buy.

The challenge for the software developer is how best to offer try-before-you-buy without having to rely on trusting someone to actually pay for the software. There needs to be a way of forcing the user to pay for the software they have been trialling. Fortunately, there are a number of strategies used by the software development community to achieve this. The more common ones include:

Limited Capability

Here the developer provides their product limited in some capability aspect. Some of the more popular strategies include limited:

- **Functionality** - menu options are greyed out to disable certain functions such as Save, Print, Export etc
- **Features** – menu options are greyed out but show greater usage and value to the user or company.
- **Reference data** – a small percentage of the information is available for use. This is typical for reference material packages such as encyclopaedias and industrial/professional material eg nutrition guides, statutory/regulatory law, policies, standards etc
- **Capacity** – only a finite number of records can be created eg 20 accounts can be created

In theory, the limitation should be sufficient to showcase the product's value but insufficient to provide long term practical use of the product. In practice, this is difficult to achieve and requires considerable thought and design. A product too limited in its capability will frustrate the user and under-represent the product. Too much capability may offer the opportunity for the user to get by and still derive some value but at no cost.



Limited Usage

If you want to test drive the All Wheel Drive V6 Turbo, then that's what should be offered. Usually a couple of 30 minute sessions is all it takes. To test drive the more prestigious cars, the dealer usually gives you use of a car for the weekend in the hope that by Monday morning you don't really want to hand it back. After all, this is a significant financial decision!

In the world of software, Limited Usage allows the developer to provide a full working version of the product but limits the usage to the following two common strategies:

- Time – where the software will operate at full capability over a predefined time frame after which the product will stop
- Tries – similar to Time but based on a predefined number of runs

In theory, the limitation should be easy to design and implement. In practice, there are significant security and architecture considerations that must be resolved. Simple tricks such as changing the clock on the PC before starting the program and resetting the time after usage will defeat Time Limited restriction. It's also relatively easy for your average programmer to debug a Tries Limited restricted program and change the count to a very large number. Alternatively, a quick search on the internet will return hacking tools or specific hacks for the program in question that removes the Limited Usage function.

Other Considerations

Start-ups and small software development businesses find themselves in catch-22 situations. To sell more software, they need to offer a demo

version. However, to fund the development, maintenance and support of the demo version in addition to the costs associated with their full version requires even more software to be sold.

The solution is to integrate Limited Usage functionality into the single version of the software package. Here the software can be distributed, installed, trialed, purchased and finally unlocked for indefinite full use. This solution requires specialised software activation technology. The issue of developing this type of technology in-house verses using activation software development kits or outsourcing to an expert service provider is a whole other topic. Some development companies believe it's easy and they can do it, some have the funding to try, while some understand their own limitations in terms of funding and expertise and believe they should focus on their core skills of developing the product while outsourcing to experts who deliver the security components because that's what they do best. Suffice it to say, 36 per cent of the software installed on computers worldwide in 2003 was pirated, representing a loss of about US29 billion dollars to companies (Survey conducted by research firm International Data Corp. for the Business Software Alliance (BSA)). Watch out if your market is Asia Pacific or Eastern Europe – piracy is running at 53% and 71% respectively.

Software Piracy is pandemic with even the most popular packages that include software activation technology suffering losses. So the decision on which try-before-you-buy strategy you adopt and how you should implement that strategy requires well considered research – it could be the difference between a suffering business or a thriving company.

Please don't hesitate to contact us should you wish to further explore these issues or provide feedback.

Saul Midler

saul@securewrap.net

Managing Director