

Refactored Cognitive Dimensions and Security Usability

Luke Church
Computer Laboratory, William Gates Building
University of Cambridge, 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK
luke@church.name

Abstract

This discussion paper considers security usability from a Refactored Cognitive Dimensions (RCDs) perspective. It outlines some benefits of doing so for an attacker and a defender, and mentions some difficulties encountered in applying RCDs to security.

Introduction

Usability is an important area of security. There have been many cases where poor usability has resulted in a system being compromised [1]. Security systems are sufficiently complex that their configuration can be considered to be a form of declarative programming which frequently performed by end users.

Modelling existing problems

To demonstrate the validity of the Refactored Cognitive Dimensions some known properties of systems will be considered from a RCDs [2] perspective:

Visibility

Microsoft Windows XP Home allows simple folder based file sharing on a local area network through NETBios shares. Whilst setting up their first file share the user is given the option of either using a Network Setup Wizard, or clicking a link reading 'If you understand the security risks but want to share files without running the wizard, click here'. They can then configure the shares on a folder by folder basis. Folder shares must have 'other users can alter my files' explicitly enabled before global write is possible.

There are several ways of observing file shares when they are configured:

- 'hand offering folder' visual clue instead of the folder icon in Windows Explorer
- Administrative Tools | Computer Management | Shared Folders | Shares
- 3rd party tools

However at the start of each term in Downing College (~200 student owned Windows XP Home machines) several students bring systems with serious file share configuration vulnerabilities. The typical response once notified is that they had changed the settings for their home network where they trusted all the users and had forgotten to reset their settings.

This is a *cognitive visibility* problem. The information is relatively easy to acquire, but it is not obvious to the user that they need to do so, and many are not aware of how to do so easily.

There is a further complexity that the Computer Management tool is intended for IT professionals and as such includes system shares that may confuse an end user.

On the other hand the 'Security Center' that was introduced with XP Service Pack 2 is intended for end users. It introduces a centralised location for reviewing the security state of the operating system [3], but includes only a small subset of the available security configuration information (specifically Updates Configuration, AntiVirus Status and Firewall status). Whilst this decreases the *structural visibility* problem by locating the information in a single place and assists with *cognitive visibility* by highlighting the issue of security state, it is not far reaching enough for even moderately sophisticated users.

As might be expected there are also configuration management tools for IT Professionals, such as the 'Microsoft Baseline Security Analyzer' [4], which provide a series of tests against the current configuration and advises of corrective actions. Whilst this tool significantly improves the *structural visibility* of the security state which may consist of many hundreds of settings, it may be too technical for end users.

Despite the security state being declaratively programmed, visibility in both its cognitive and structural forms remains a problem in perceiving system security state.

Viscosity

Much of security can be viewed as management of viscosity. The aim of many systems is to maximise the *structural viscosity* for the attacker and minimise it for a legitimate user of the system.

Viscosity Management

For example, the common security principle of running as least privilege is a viscosity tradeoff. A user running as Administrator/root can typically perform any action on the system with minimum interference, and hence low viscosity due to security effects. However an attacker who manages to gain control of the account experiences the same lack of viscosity to harmful actions.

Running as a lower privilege account results in some actions requiring privilege modification, adding *cognitive viscosity* (the action now consists of a privilege modification and an action) for the end user, but adding more viscosity to the attacker who is hopefully unable to perform the privilege modification.

Runas and Sudo can therefore be viewed as viscosity management tools from a user perspective.

Viscosity Avoidance

Viscosity has a further serious effect on security. If a system's security features offer too much viscosity to the typical usage patterns of even a well motivated user, they will tend to be disabled 'to get the job done'. There are many examples of this occurring, including very high profile cases [5].

As discussed later, this property of viscosity can be exploited to both the attacker and the defender's advantage in different cases.

Discriminability

Discriminability is a cognitive dimension that is frequently abused in phishing attacks. The victim is often fooled into believing that a website is from a different organisation than it pretends to be through a combination of techniques to reduce the discriminability. These range from using visually similar URLs (e.g. www.PPIG.org, the 'P' is UTF U+03A1, instead of www.PPIG.org, the 'P' is UTF U+0050) to duplicating stylistic layouts of the site. [6]

The anti-phishing tools largely work by using technical measures to increase discriminability. [7]

Attack and Defence in Usability Space

As well as modelling existing systems, an RCD analysis gives rise to the possibility of specifically targeting the usability of a system rather than its information structural properties.

Viscosity attacks

A *viscosity manipulation attack* manipulates the viscosity of the system in favour of the attacker. For example, if a hostile application trying to talk through a bidirectional software firewall can cause the firewall to repeatedly query the user for instructions to allow/deny the application, then the *cognitive viscosity* of the system will be substantially increased.

As has been seen before, a typical response is to decrease the security level in order to decrease the number of requests, allowing the hostile application to communicate out the information it wishes to. This is analogous to a museum curator switching off a burglar alarm that has given many false alarms in a row, only to find that the circumstance had been manufactured for the benefit of a real criminal action. [1]

Hidden Dependency Optimisation

Alternatively viscosity manipulation can also be used by a defender in 'soft-security' situations to make it easier to use a system, provided it is done in a particular chosen manner. Use of the system in any other manner will result in sufficiently high viscosity that the user is likely to give up.

This can sometimes be achieved by manipulating the dependencies to create *structural viscosity*. For example a music file could be split up into many streams, with an arbitrarily complex encoding scheme. The functionality to play this back is all encoded in a provided piece of playback software. The owner of the music wishes to make it difficult to break the music played down into its constituent tracks.

Presented with such a system, the task of decoding the music to the played back streams may be a non-trivial one. The may choose to pursue either:

The intended low viscosity route: Play the music with the supplied software

The high viscosity route: Reverse engineer the file format, construct a stream analyser and mixer and separate the file into its constituent streams.

Whilst this does not offer strong security it may be adequate for the purpose, if the value to the attacker of the high viscosity route is sufficiently low.

Hence the developer has modified the *structural viscosity* of the system to strongly promote the desired usage behaviour, dependent on the context, this may be enough to achieve the desired result.

Difficulties with RCDs and Security Usability

However, despite RCDs being useful for modelling existing systems and developing new systems, there are some issues that are not easy to represent within the RCDs framework.

- Acceptability: What cognitive dimension is associated with the difficulties of blood sample authentication?
- Motivational problems associated with security being a secondary task [8]
- Visual effects of metaphors and grouping are problematic to represent
- Temporality causes difficulties in modelling 'well in advance' knowledge models [8], though RCDs may help this through increased structural modelling

Whilst these are currently limitations, it seems that RCDs provide a potentially valuable tool in modelling of security usability issues.

Conclusion

RCDs provide a tool that can be used to model some elements of security usability. It shows promise for revealing methods of manipulating the cognitive properties of a system, however there seem to be some properties that are difficult to model with RCDs.

References

1. Anderson, R. (2001) Security Engineering, Wiley Computer Publishing
2. Green T. R. G., Blandford A. E., Church L. Roast C. R. What CDs did next (Pending)
3. http://www.microsoft.com/windowsxp/using/security/internet/sp2_wscintro.msp
4. <http://www.microsoft.com/technet/security/tools/mbsahome.msp>
5. United States Senate Select Committee on Intelligence, CIA Office of Inspector General Investigations Staff Report on the Improper Handling of Classified Information by John M. Deutch, 106th Congress, <http://intelligence.senate.gov/igreport.pdf>
6. Howard, M. LeBlanc. D. (2003) Writing Secure Code, Second Edition, Microsoft Press
7. Miller, R. C., Wu M. (2005) Fighting Phishing at the User Interface, in Security and Usability, O'Reilly Press
8. Whitten A., Tygar J.D. (1999) Why Johnny Can't Encrypt: A Usability Evaluation of PGP, 8th USENIX Security Symposium