Usability Through the Stack

Angela Sasse

Mary Ellen Zurko

- How can we do system level analysis of security protocol ceremonies to include people (level 8)? What does that teach us? What about the social level (9)? And the geopolitical level (10, or 1)?
- What would we learn or could we do in hcisec by unabstracting our layers?
- How do we handle lower layer security errors to ask the user appropriate questions in the application context?
- HCISEC Metaphors – design lower layers to support them? Push them down from the upper layers? Develop a specific metaphor for security errors (monsters under the bed)?
- Dealing with varying HCISEC model requirements – multiple models, or multidimensional models? What about conflicting control assumptions? Who resolves the conflict?

- How would designing for HCISEC change the abstractions in our layers?
  - Can we still provide useful HCISEC by tweaking the abstractions we have?
  - How do we get robustness in the face of malevolence?
- What is relevant to HCISEC that our protocols can provide? The name of the certificate at the other end of the connection is not the answer.
  - What does the user need to know?
  - What does the user want to know?
- How do the semantics of mechanisms such as access control effect the end user experience, and how should they?

- What are the easy to use security mechanisms? What makes them attractive? How can we make the right ones as easy to use?
- What information is needed from lower levels to interact with the user (on security errors)?
- How do we help application developers at upper levels understand and use the security information from lower levels?
- What incentives will help the lower levels send back the right information for HCISEC, and the upper levels consume it?

- What API metaphor can structure developer thinking to get over the impedance mismatch between the user and stack models?
- Are there ways to add security and privacy to lower layers that are wins and don’t impact the user?
- If we could design a user level authentication of a server, could we design a system to ensure that it could not be stolen?
- What new ambitious HCISEC problems could we design the stack to support?
- We have the languages to let users specify whatever policy they want. What are the real HCISEC challenges? Provisioning? Control and error questions?

- What if we change the abstractions, say from hosts in the network to user data. How would we express protocols in those terms? Would this help with user control of their information? Does moving the security abstractions to the data make it safer?

- How can SSL and other forms of (user) security be made more pervasive?

- What are the HCISEC implications of user lifecycle of data? Who is it going to, who can store it, what is the encryption, how is it controlled? Provenance, availability?

- What kind of burden would/does it place on users to manage their own policies?

