Comments on Security

P.B. Ladkin

Use of resources requires

Responsibility:
Users adhere to *conventions* of use

Conventions may be
- legal
- contractual
- social

Accountability:
Actions and states (data, permissions) are attributable to individuals

Data is	file contents message contents
Permissions are	who may read who may write who may act

Examples

Obligations:

Legal	don't threaten others don't impersonate others ? don't hinder others' work ?
Contractual	don't play games don't write personal letters keep business contracts private
Social	porno to be read-protected don't read others' mail

Access	to data	(files, messages, infoflow)
	to state	(system state)
	to actions	(read, write, do)

Access control:

Responsible group

Permitted group

Unix:

everything is a ‘file’

responsible	=	*user*
permitted	=	system-defined group
		+ ‘*superuser*’
actions	=	read, write, do

Accountability:

User + Superuser

Superuser can do anything
Much system data only writeable by superuser
Some system data only readable by superuser

Authentication follows *public-key* scheme
(although public-key identified after Unix built)

Key management under user control

Problem:
users need to perform certain sysoperations

Example:
public keys contained in one file, */etc/passwd*
Authentication easy: compare E(K,passwd)
with that in */etc/passwd*

however:
Key management requires user to write */etc/passwd*
so either

- all users must have write access
- or users must ‘spoof’ *superuser*

First option infeasible
(users may write others’ passwords,
achieving *denial of service*)

Second option taken: *setuid* command

Potential security hole:

a malicious user runs arbitrary processes

- after a process has executed *setuid(root)*
- before it has executed *setuid(user)*

Problem:

- there is only one level of privilege: *do all*
- programs which *setuid* are not provably secure

Alternative design options:

- important control files individual to permit-group
- different security levels for services
- no pervasive trusted-user privilege
- rigorous trusted-user authentication
- no ID change mechanism

Internet communication problems

- keys sent in cleartext: vulnerable to
  - physical snooping
  - system holes allowing user snooping
- E-mail not reliably authenticated
- denial of service
- nuisance communication
- local breaking and entering
- masquerading

These were not problems when

- nets were physically secure
- privileged users
  - could be identified
  - could be trusted
  - configured trusted mail
  - controlled authentication

Encryption of restricted information solves many of these problems: for example

Channel communication via tunnelling

Tunnelling requires encryption after X
$M \mapsto E(K, M)$
and decryption before Y
$EM \mapsto D(K', EM)$
where $D(K', E(K, M)) = M$

Encryption Functions E, D

Capability:

$E(K, M) \mapsto \langle K, M \rangle$ hard
$D(K', EM) \mapsto \langle K', EM \rangle$ hard
That is, E and D are *one-way*

Feasibility:

$K, M \mapsto E(K, M)$ should be easy
$K', EM \mapsto D(K', EM)$ should be easy

Secrecy:

$E(K, M) \mapsto M$ should be hard
$D(K', EM) \mapsto K'$ should be hard, for every EM

Authenticity:

$E(K, M) \mapsto K$ should be hard, for every M
$D(K', EM) \mapsto EM$ should be hard

One-key Systems

Capability + Feasibility + $(K = K')$

If K *known*,
$E(K, M) \mapsto K$ easy
$D(K, EM) \mapsto K$ easy

Consequently
No Secrecy
No Authenticity

If K *not known*,
$E(K, M) \mapsto K$ hard
$D(K, EM) \mapsto K$ hard

Consequently
Secrecy
Authenticity

Conclusion:
good for *single user* protection
not good for *multi-way* (how is K shared?)

Policies

Mail:
Message contents should generally be secret
Signatures should generally be authentic

Other Transactions:
Generally secret and obscure

Two-key Systems

Diffie-Hellmann 1976: public-key encryption

good for multiway, e.g., communication

Capability + Feasibility + ($K \neq K'$)

Secrecy:
K public, K' private
$M \mapsto E(K, M)$ easy
$EM \mapsto D(K', EM)$ hard
Sending easy, receiving hard

Authenticity:
K private, K' public
$M \mapsto E(K, M)$ hard
$EM \mapsto D(K', EM)$ easy
sending hard, receiving easy

Everyone has both private and public keys

The Social Dimension of Cyberthreats

Hazard - system state that, in concert with certain (worst case) environmental conditions, leads inevitably (Leveson) to an accident

Accident - undesired, unplanned (not necessarily unexpected) event resulting in a specified level of loss

Threat - similar to hazard but *intentional*

Technically, there's little to choose between threat and hazard

Socially, there is a world of difference

There are technical consequences, however

Intentional threats are **persistent**

Intentional threats are **goal-seeking** (teleological)

See Searle, *The Construction of Social Reality* for the 'technical' construction of social conventions.

To counter cyberthreats we need to know:

what is the *intention*?

what are the *social conventions*?

how are these conventions implemented?
(technically, socially)

we then want to limit system behavior which leads to **persistent, teleological accidents**

Can it be done?

Lots of social institutions don't fail in quite as disastrous a manner as automated, networked institutions.

What's the difference?

Adherence to convention, encouraged by **accountability**

We 'technical' experts seem to be weak on the implementation of accountability, as well as on other social features. I would suggest that that's where we need to focus