The Inevitability of Failure:

The Flawed Assumption of Security in Modern Computing Environments

Peter A. Loscocco, *Stephen D. Smalley*, Patrick A. Muckelbauer,
Ruth C. Taylor, S. Jeff Turner, John F. Farrell

tos@epoch.ncsc.mil

National Security Agency

Agenda

The Inevitability of Failure

Deja Vu

Missing Links

Flawed Assumptions

Meeting the Need

No Other Way

Reaching the Goal

The Inevitability of Failure

The Flawed Assumption

- Adequate security can be provided in applications with the existing security mechanisms of mainstream operating systems.

Our Thesis

- Secure applications require secure operating systems, and any effort to provide system security which ignores this premise is doomed to fail.

Deja Vu

Doesn’t everyone know that?

- Many people know it, but aren’t doing anything about it.
- Some people are actually suggesting that it is false.

Why revive the issue now?

- More critical today due to increased connectivity, sharing
- Increasing dependence of critical services on computing
- Greater awareness of need for security among end users
- Misguided security efforts are prevalent in industry and academia

Deja Vu

Haven't we tried before and failed?

- The operating system community never truly tried.
- Security practitioners failed to demand secure operating systems.
- TCSEC distorted the concept of a secure operating system.

Isn't the benefit outweighed by the cost?

- Not for any environment where security is a requirement.

Don't we already have secure operating systems?

- Yes, but they aren't mainstream and they ignore or lack critical functionality.

Missing Links

Assurance

- demonstrate that the features meet the desired properties
- demonstrate that the features are implemented correctly
- critical for demonstrating that features operate correctly, but still need right set of features for security

Features

- mandatory security, trusted path, protected path
- critical for meeting properties, but undependable in the absence of assurance evidence

Mandatory Security: Definition

Policy tightly controlled by policy administrator

Not just "global and persistent" policies

- although supporting such policies is important
- MLS, Biba, Clark-Wilson, type enforcement, dynamic RBAC
- our research in flexible policy support: DTMach, DTOS and Flask
- *The Flask Security Architecture: System Support for Diverse Security Policies*
- http://www.cs.utah.edu/projects/flux/fluke/html/flask.html

Not just access control policies

- cryptography, authentication, network security services

Mandatory Security: Key Gains

Provides critical support for application security

- protects against tampering with secured application
- protects against bypass of secured application

Provides strong separation of applications

- permits safe execution of untrustworthy applications
- limits scope of potential damage due to penetration of applications
- functional uses: isolated testing environments or insulated development environments

Protects information from

- legitimate users with limited authorization
- authorized users unwittingly using malicious applications

Trusted Path and Protected Path

Trusted path

- permits a user to directly interact with trusted software and cannot be imitated by other software
- prevents malicious applications from impersonating trusted software to the user or vice versa

Protected Path

- mutually authenticated channel between software components
- ensures that critical system functions are not being spoofed
- OS can provide simpler, more efficient support directly

Flawed Assumptions

Application-space cryptography

- the calling application cannot be bypassed or modified
- only authorized components may serve as cryptographic components
- only authorized applications may use the cryptographic component
- the initial keys were securely established in the cryptographic component
- only authorized access to the cryptographic component's code and data is granted

Other examples provided in our paper:

- general case of application-space access control
- mobile code, network security protocols, Kerberos, firewalls

Meeting the Need

Mandatory security

- can ensure that calling application is unbypassable, tamperproof
- can protect against use of unauthorized cryptographic component
- can isolate cryptographic component from other users' applications or untrustworthy applications executed by authorized user
- can protect initial keys and cryptographic component's code and data

Trusted Path

- can protect establishment of initial keys
- can protect against misuse of cryptographic component

Protected Path

- can ensure that malicious software cannot impersonate cryptographic component
- can enable cryptographic component to identify clients

No Other Way

Discretionary security

- little protection against legitimate users with limited authorization
- no protection against malicious applications
- places burden of security on end users

"New" security paradigms

- many ignore the well-established fact that applications require a dependable infrastructure
- some recognize need for infrastructure, but have no plan to achieve it
- offer no satisfying replacement for the "old" paradigm

Reaching the Goal

Security practioners

- must openly acknowledge their security solution’s assumptions and the corresponding risks to consumers
- must state their security solution’s operating system dependencies as requirements for future operating systems

Consumers

- must demand secure operating systems from vendors

Operating System Vendors

- must demand secure operating system technology from researchers
- must integrate existing technology into products

Operating System Researchers

- must initiate new research in secure operating systems

Reaching the Goal Part Deux

This will lead to commercially viable secure systems

- thereby permitting security practitioners to concentrate on the security services which belong in their particular components
- thereby freeing them from trying to address the total security problem in their components, which is doomed to fail.

Questions?

tos@epoch.ncsc.mil

http://www.cs.utah.edu/projects/flux/fluke/html/flask.html