Presentation for “Classic Papers” track at ACSAC 2007, Miami Beach, December 11–14, 2007

Distributed Secure Systems: Then and Now

Brian Randell
School of Computing Science
Newcastle University
Newcastle upon Tyne UK

John Rushby[a]
Computer Science Laboratory
SRI International
Menlo Park CA USA

[a] At Newcastle when this work was done

Overview

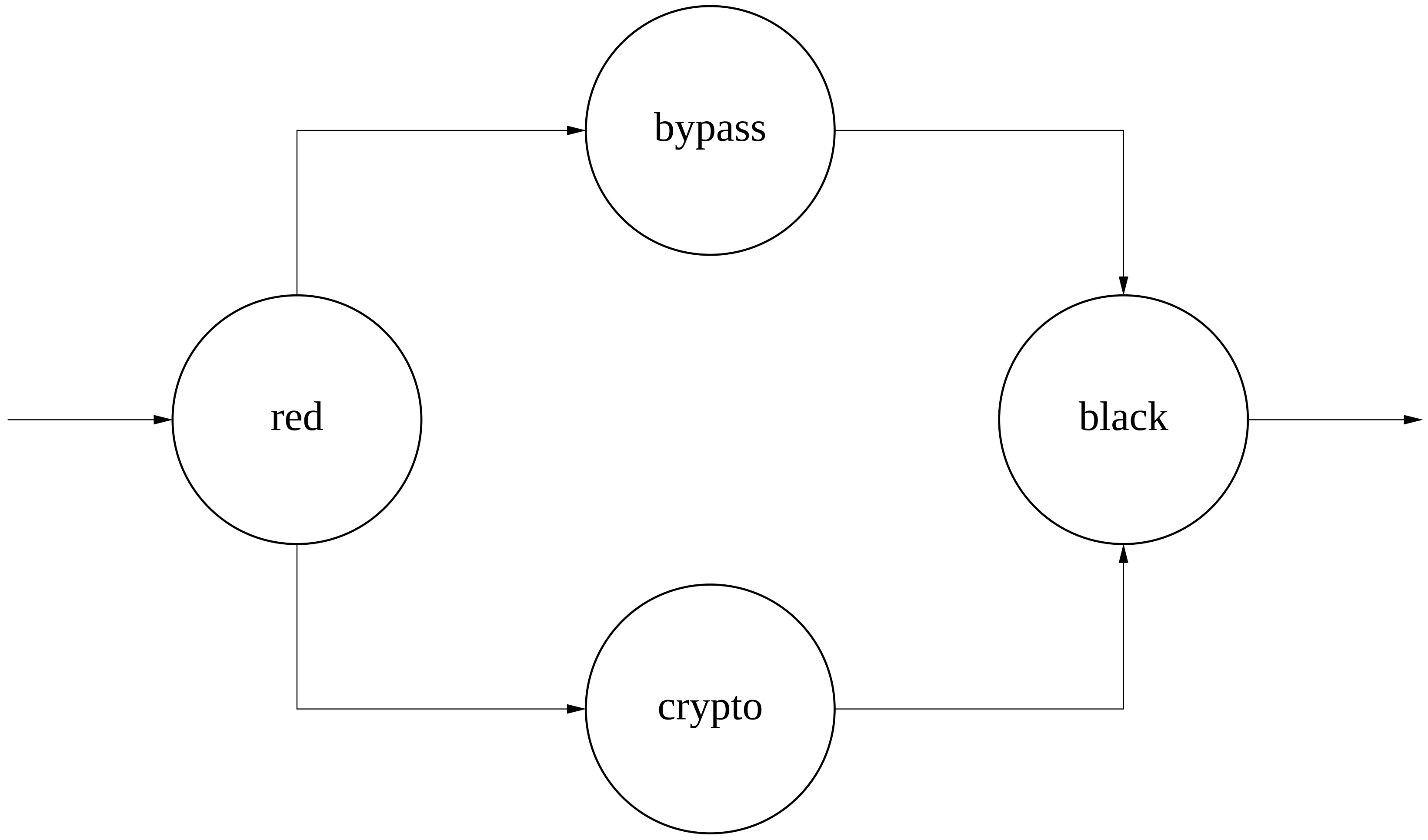
- 1979–1983: History and reminiscence
    - Security
    - Distributed Systems
    - The Distributed Secure System (DSS)
- 1984–1994: Subsequent developments
- 1995–2006: Interregnum and rediscovery
- 2007–: Looking forward

Security: 1979

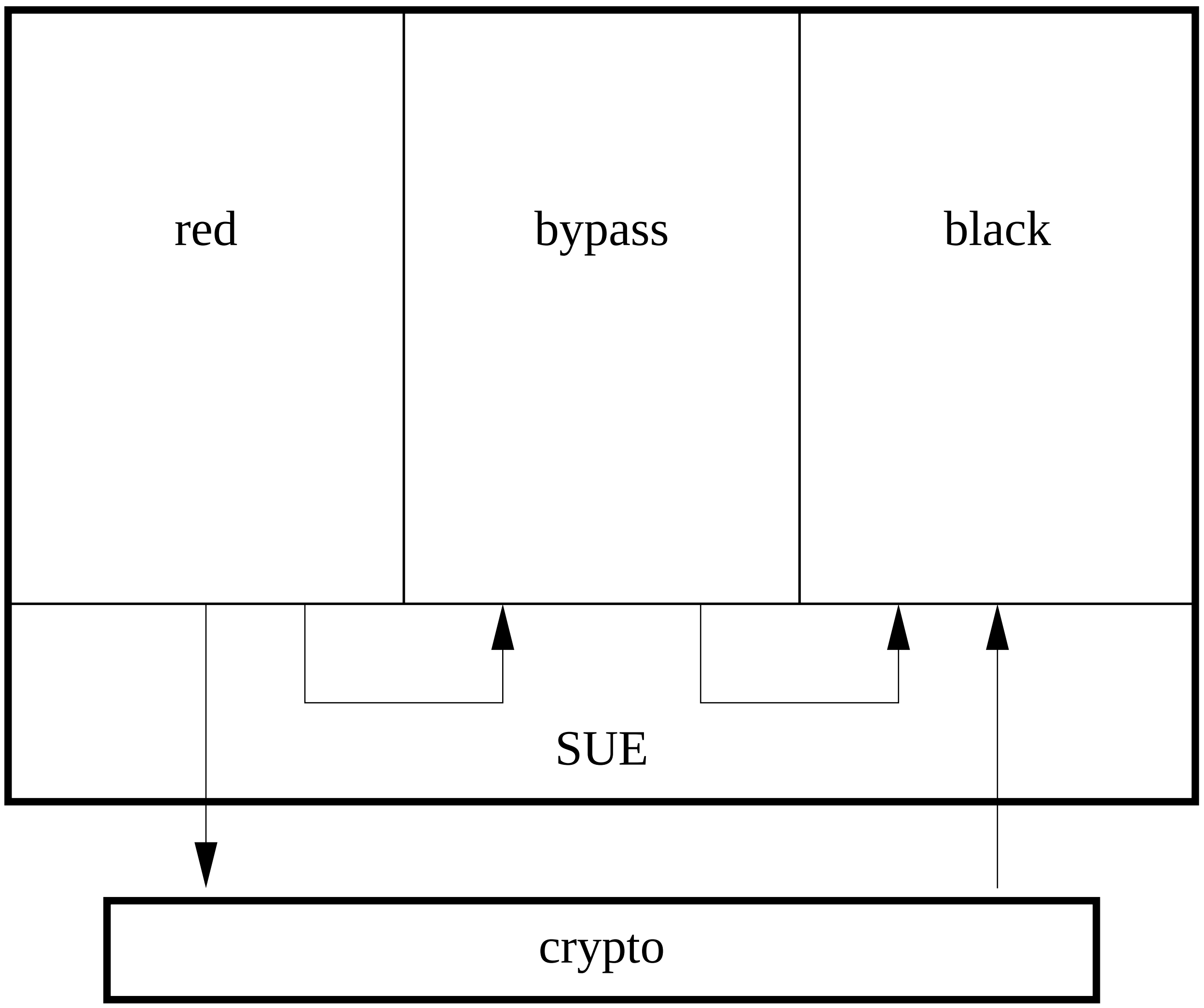
- The UK Royal Signals and Radar Establishment (RSRE)
  - Later part of Defence Research Agency (DRA), and partially privatized as QinetiQ

  Had developed a secure "Pilot Packet Switched Network" (PPSN)
- Used end-to-end encryption
- With the encryption functions performed by "Packet Forming Concentrators" (PFCs)
- Which were minicomputers that used a "Secure User Executive" (SUE) to enforce red-black separation

Red-Black Separation



Red-Black Separation in the PFCs



Newcastle Research Contract

- RSRE were interested in issues of assurance and certification for the SUE and the PFCs
- Were aware of US work in formal modeling of security, of formal verification applied to security, and of precursors to the Orange Book
- Funded a research project at Newcastle (led by Peter Henderson) to explore these topics, and US approaches and technology
- Rushby joined the project (from Manchester, where he was a lecturer) as its (only) Research Associate
- RSRE also funded consultancy with SDC

Security Orthodoxy circa 1980

- The Anderson Report had identified the central importance of "reference mediation"
- To be performed by a "reference monitor"
  - A component that ensures that all data references are in accordance with policy
  - Tamperproof, nonbypassable, and correct
  - Credibility and feasibility of strong assurance for correctness suggests the reference monitor should be small and simple
- Policy was usually the Bell and La Padula model of MLS
- And the reference monitor was identified with a customized operating system kernel
  - These became known as "security kernels"

The SUE as a Security Kernel

- Over the course of several extended visits to the US
  - E.g., SDC, ISI, UCLA, SRI, Purdue, U Texas, MIT
  - Received everywhere with unfailing courtesy and generosity and willingness to share

  Rushby began to realize

  1. The SUE is not easily interpreted as a classic security kernel: it is not the sole arbiter of policy
  2. Maybe the approach used in the SUE and PFCs is preferable to the orthodox approach, at least for embedded systems and network components
- Led to Design and Verification of Secure Systems, SOSP 1981

Security Composed Of Many Small Policies

- Putting policy in the kernel is fine when there's a single policy
- But what about cases where the overall security argument requires cooperative composition of several different policies?
- E.g., PFC requires red-black separation (no direct channel from red to black), bypass trusted to reduce leakage to acceptable level, crypto trusted to do strong encryption
- Even MLS systems have components that are trusted to do specialized functions, such as login authentication
- And sometimes those components have to violate the basic MLS policy—the problem of "trusted processes"
- And sometimes the whole purpose of a system is contrary to the basic tenet of MLS, such as "guards" (trusted high to low filters), so building it on an MLS kernel seems perverse

Separation

- The 1981 SOSP paper argued that it is seldom appropriate to enforce policy in a kernel
- Instead, the kernel should create an environment in which it is feasible for many separate policy components (reference monitors) to coexist and cooperate
- That environment is a conceptually distributed system with known components (boxes or circles) and known communication paths between them (arrows)
- No interaction between one component and another except via the known channels
- The key properties of this environment are separation (of components and channels) and enforcement of the configuration (the "wiring diagram") specified for them
- Similar to partitioning in avionics/safety

Separation Kernels

- Note: a channel from A to B doesn't mean A can make arbitrary changes to B's state (e.g., write anywhere in its memory)—that would make implementation of reference monitors impossible—changes are to a known localized part of B's state/memory (called a port or buffer)
- An operating system kernel that enforces separation and configuration integrity is called a "separation kernel"
- The SUE was a separation kernel

Two-Level Security Architectures: 1981

Separate the issues of policy from those of resource sharing

1. Conceive of the system and its policy enforcement as a suitable configuration of separated components and channels
   - Conceptually, a distributed system
   - Abstractly, a circles and arrows picture
   - With trusted reference monitors in some of the circles
   - The absence of an arrow is often particularly important
     - E.g., no direct arrow from red to black
2. Use a separation kernel to implement this conceptually distributed system in a single machine

Note:

Assurance that composition of the policies enforced in the trusted components yields the desired overall security objective was left as an exercise for the reader

Distributed Systems: 1979

- Local area networks were becoming available
- And small minicomputers (PDP-11s) were fairly inexpensive
- So you could build a network of workstations
- But how would you actually organize them for distributed computation?
  - Business as usual (FTP, telnet, email)
  - Distributed file system (e.g., NFS)
  - A true distributed system (e.g., Locus)
- Newcastle had a long-standing program in reliability and fault tolerance (led by Randell) and was interested in using distributed systems to mask faults in computations
- Looked for existing distributed system foundation, but came up with a better one of their own

The Newcastle Connection and Unix United

- Lindsay Marshall invented a layer of what would now be called middleware ("The Newcastle Connection") to extend the hierarchical file system of a single Unix system across a network of such systems ("Unix United")
- Extend the namespace above `root`, so that `/../unix2/home/brian/a` names a file `a` on another machine (called `unix2`)
- If `a` is a program, we get remote execution, and if it is data we get remote file access
- The Newcastle Connection middleware intercepted system calls and redirected those requiring remote execution or file access using remote procedure calls

Two-Level Security Architectures: 1982

- In 1981, distributed systems were the conceptual model for the upper level, but the implementation was a logical simulation, using a separation kernel to recreate the security attributes of the physically distributed ideal
- Now, with Unix United, it became feasible to realize the conceptual model directly
- Except that would be wasteful for small components
- So you'd want a combination
- But there are further ways to realize separation apart from logical and physical
  - temporal: classic periods processing
  - cryptographic: encryption and checksums
- Could imagine using all four mechanisms in a single system

The Distributed Secure System (DSS)

- The DSS was a two-level security architecture that used all four separation mechanisms to create an MLS system
  - Physical separation for servers of each classification
  - Crypto separation on the LAN and to create a shared file system that used a single backend server
  - Logical separation in the controllers for these
  - Temporal separation for single-user workstations
- Called The Distributed Secure System rather than Secure Distributed System to stress that is was a secure system that used distribution to achieve the goal
- It appeared as a single coherent system despite its distributed and separated implementation

Subsequent Developments: 1984–1994
The UK DSS Technology Demonstrator Programme

- RSRE started a “Technology Demonstrator Programme” (TDP) to develop prototypes of DSS
- The first TDP in IT (usually they were tanks or ships)
- The present authors were not involved
- Emulation in 1985 “demonstrated full internal functionality of the DSS” with applications aimed at office automation
- Good progress on full DSS reported in 1991, aimed at Level 5 in the “computer security confidence scale” then used in the UK (roughly B3 in the Orange Book)
- Actually awarded Level 4 in 1993 and insertion trials undertaken at three sites in 1994

DSS TDP Insertion Trials: 1994

HQ PTC Innsworth: first attempt failed due to errors in crypto keys provided by CESG; second attempt hampered by bad Ethernet interfaces; considered too slow for regular use, and unreliable

DRA Fort Halstead: failed due to networking problems (missed key packets under heavy load)

HM Treasury: abandoned due to problems in first two trials

- Fixes to the problems in reliability and performance would require "significant reengineering of the DSS kernel"
- "It is unlikely that MOD or DRA will provide further funding for DSS development. . . its future therefore depends on the licensees being convinced that the necessary substantial investment will be worthwhile"
- The two commercial licensees presumably abandoned it

Interregnum

- A decade of effort led to a disappointing failure
- What can we say?
- Naturally, we tend to attribute the problems to technical limitations of the time, pioneering use of middleware and distribution, and to UK development and management practices
- So we remain serene and confident in the rightness of the DSS ideas
- Modern Integrated Modular Avionics (IMA) systems are similar and are deployed successfully

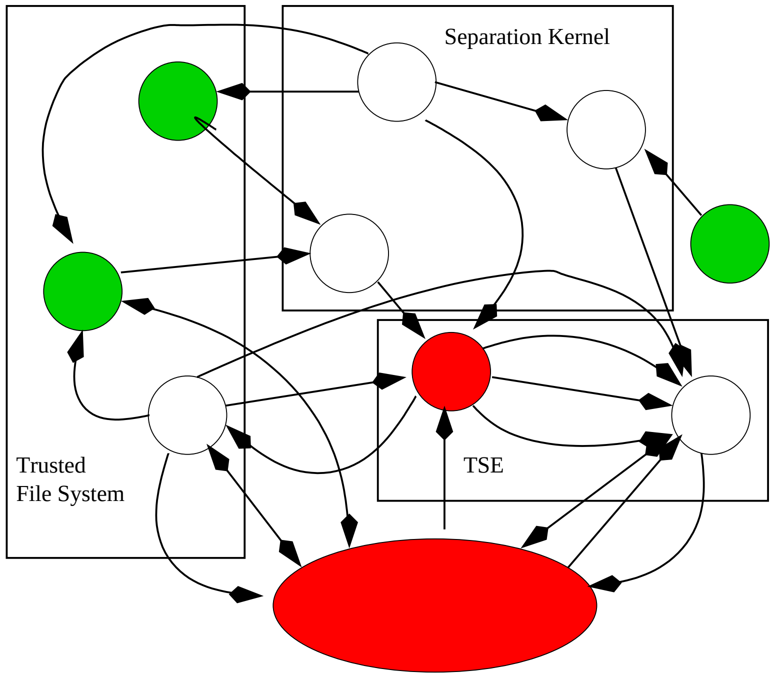
Rediscovery: 1990s

- The US had seen disappointments of its own in secure system development
- This led to reconsideration of monolithic security kernels, and renewed interest in separation kernels
  - “In 1993 an informal separation kernel working group was established” at NSA
- Rather later, an architecture called MILS emerged (Vanfleet and others 1996, 2003; Alves-Foss and others 2004, 2006)
- These usually cite SOSP 1981 as their inspiration, but we think DSS (and “Networks Are Systems” from 1986 “Red Book” workshop) are closer to these later architectures

Two-Level Security Architectures: 21st Century

- Current systems are large and complex, but now we have mature middleware, and effective development and management practices
- Try to arrange the upper (circles and arrows) level so that security depends on only a few trusted components
- And those are trusted to do only relatively simple things
- Split big components up if necessary to achieve these
- We can afford to have lots of circles and arrows, and should use this to reduce and simplify the trusted components
- We can afford lots of circles and arrows because the MILS lower level can efficiently provide fine-grained separation of physical resources

Two-Level Security Architectures: The MILS Approach



Care and skill needed to determine which logical components share physical resources (performance, faults)

Top-Down and Bottom-Up

- DSS was purely top-down: the lower-level components were engineered for their specific role
- But MILS aims to foster a competitive COTS marketplace for lower-level components
  - So needs to identify a useful set of separated/partitioned resources and services
  - And wants the components to be pre-certified
  - (HAP has a similar architecture and motivation, but aims for a common platform rather than components)
- So MILS requires a design approach that is partly top-down, and partly bottom-up (to use existing components)

Assurance and Certification in MILS

- Recall, in DSS

  Assurance that composition of the policies enforced in the trusted components yields the desired overall security objective was left as an exercise for the reader

- MILS was in this position, too
- And our paths crossed in 2005
  - Rushby had spent most of the previous 25 years involved in development of formal verification tools, and their potential application in (aircraft) certification
  - Had been a member of the committee that generated IMA guidelines DO-297
- For MILS, we need an approach to system certification based on separately certified components: compositional certification

Looking Forward: 2007–
Compositional Assurance for Safety and Security

- Long term, a shift to goal-based assurance is needed
- For now, in security, work with the Common Criteria (CC)
    - Which is specialized though Protection Profiles (PP) to Security Targets (ST) to a Target of Evaluation (TOE)
- MILS is a two-level security architecture
- Trusted components of the upper level are called operational
- Trusted components of the lower level are called foundational
- We need protection profiles for these two classes of components
- And a MILS Integration PP (MIPP) to tie it all together
- This is joint work with Rance DeLong

Two Kinds of Components, Two Kinds of PPs

The foundational and operational levels of the MILS architecture have different concerns and are realized by different kinds of components having different kinds of PPs

Operational level: components that provide or enforce application-specific security functionality and policy

- Examples: downgrading, authentication, MLS flow
- Their PPs are concerned with the specific security function that they provide

Foundational level: components that securely share physical resources among logical entities

- Examples: separation kernel, partitioning communication system, console, file system, network stack
- Their PPs are concerned with partitioning/separation/secure sharing

Two Kinds of Components, Three Kinds of Composition

We need to consider three kinds of component compositions

operational/operational: need compositionality

foundational/operational: need composability

foundational/foundational: need additivity

Take these in turn

Compositionality

Operational components combine in a way that ensures **compositionality**

- There's some way to calculate the properties of interacting operational components from the properties of the components (with no need to look inside), e.g.:
    - Component A guarantees P if environment ensures Q
    - Component B guarantees Q if environment ensures P
    - Conclude that $A || B$ guarantees P and Q
- Assumes components interact only through explicit computational mechanisms (e.g., shared variables)

Composability

Foundational components ensure **composability** of operational components

- Properties of a collection of interacting operational components are preserved when they are placed (suitably) in the environment provided by a collection of foundational components
- Hence foundational components do not get in the way
- And the combination is itself composable
- Hence operational components cannot interfere with each other nor with the foundational ones

Composability makes the world safe for compositional reasoning

Additivity

Foundational components compose with each other **additively**

- e.g., **composable(kernel)** + **composable(network)** provides **composable(kernel + network)**
- There is an asymmetry: partitioning network stacks and file systems and so on run as clients of the partitioning kernel

Near-Term Plan

- Flesh this out to yield a mathematical model for MILS and its approach to compositional certification
- And an informal interpretation (“The MILS Constitution”)
- And a formulation appropriate to a PP
- Seek buy-in: academics, developers, integrators, evaluators, government

The Longer-Term Plan

- There is general unease with software and systems (un)dependability, and emerging consensus on an approach (NRC Report, Daniel Jackson)
- There are massive recent advances in the power of automated verification and beginnings of industrial takeup (VSI, Shankar)
- There is disquiet at the costs of standards-based certification and doubts about its efficacy under modern business and development practices (COTS, outsourcing, reuse, composition, runtime adaptation)
- So it's time to develop a Science Of Certification
  - See my three papers of 2007

  And a supporting technology
  - E.g., The Evidential Tool Bus (see my 2006 paper)

Conclusions

- It is generally accepted that it takes about 25 years for a research idea to find its way into practice
- Be patient
- Be early
- "It is a great advantage for a system of philosophy to be substantially true" [George Santayana]
- Be . . .

Thanks (from John Rushby)

- Too many to name over a period of 30 years, but particularly:
- Brian Randell, Tom Anderson and others at Newcastle
- Derek Barnes and others at RSRE
- Tom Hinke and others at SDC, SRI, ISI, U Texas who introduced me to security and mechanized formal methods
- My colleagues at SRI, particularly Peter Neumann, Pat Lincoln, N. Shankar, Sam Owre, Bruno Dutertre, Ashish Tiwari, Friedrich von Henke
- Ricky Butler and others at NASA Langley
- MILS people
- And those who sponsor, guide, and participate in the MIPP effort: Jahn Luke, Tod Reinhart, Wilmar Sifre, and Dilia Rodriguez of AFRL, Carolyn Boettcher of Raytheon, and Rance DeLong (LynuxWorks)