An Appreciation of Some of
Brian Randell's Contributions To Computer Security

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

Prelude

- Brian joined Newcastle in 1969
- I was an undergraduate student at Newcastle 1968–1971
  - Brian taught an operating systems course
- And I continued as a PhD student 1971–1974
  - Brian started the Reliability Project
  - And Systems Research Group seminars
- And I returned as a Research Associate 1979–1983
  - I worked with Peter Henderson and, later, Brian on Computer Security
- Although Brian has made many contributions to computer security, I'm going to talk mainly about the work I was involved in during the early 1980s, and its subsequent history

Overview (for that part)

- 1979–1983: History and reminiscence
  - Security
  - Distributed Systems
  - The Distributed Secure System (DSS)
- 1984–1994: Subsequent developments
- 1995–2010: Interregnum and rediscovery
- 2011–: Looking forward

Security: 1979

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

  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
- RSRE were interested in issues of assurance and certification for the SUE and the PFCs
- They funded a research project at Newcastle
  - Led by Peter Henderson, staffed by me

  to explore these topics

Security Orthodoxy circa 1979

- 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
- The reference monitor was identified with a customized operating system kernel
  - These became known as security kernels

What's in a Security Kernel?

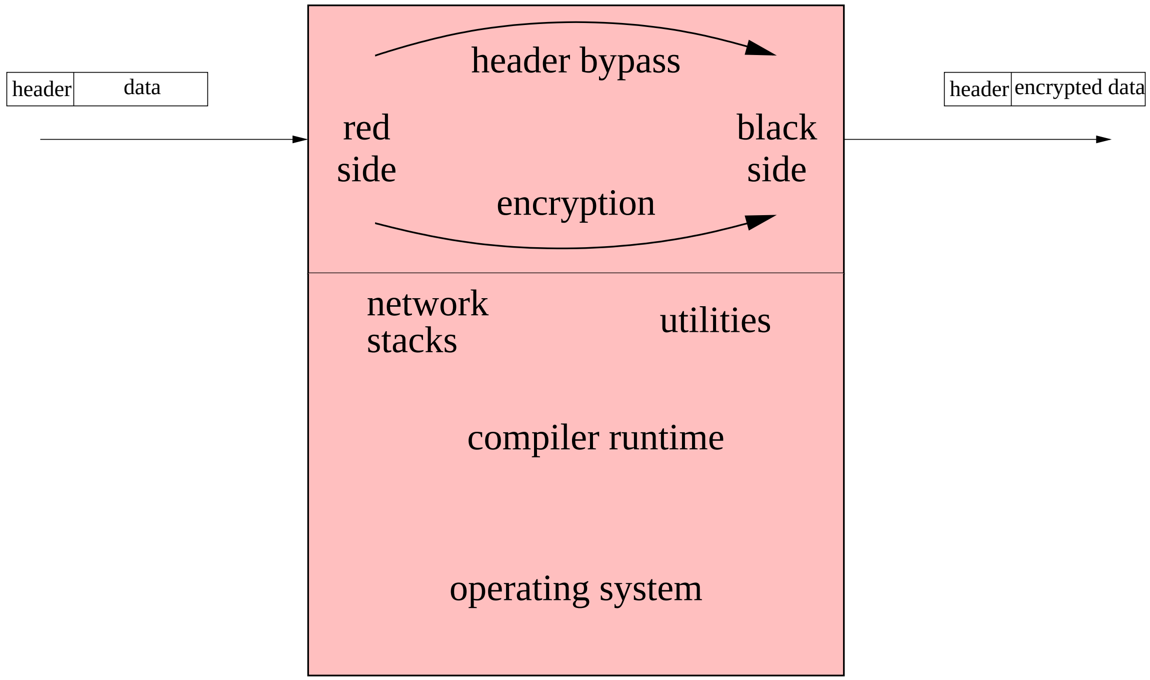
- Classical OS kernel functions
  - Process isolation, IPC, memory management, etc
- And security policy enforcement
  - Usually military multilevel security (MLS)
  - Information can flow from SECRET to TOP SECRET, but not vice-versa
- And all the other trusted functions
  - Authentication, login etc.
- And all the mechanisms to bypass policy
  - Downgraders etc. (now called Cross Domain Solutions)
- That's a lot of stuff!

The SUE as a Security Kernel

- The SUE is not easily interpreted as a classic security kernel: it is not the sole arbiter of policy
- What it does is Red-Black separation
- So that the bypass and the crypto can enforce policy

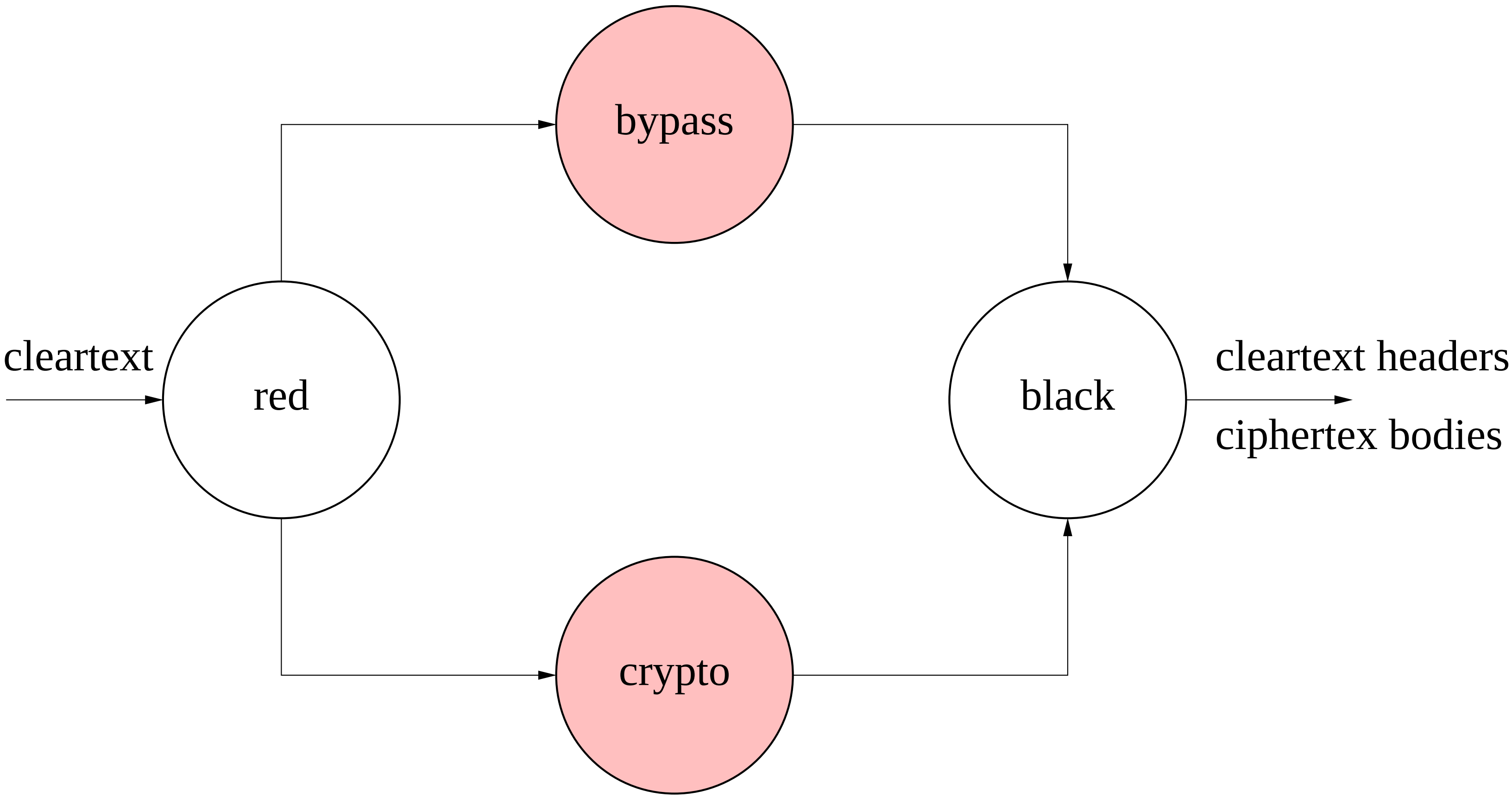
Red-Black Separation (Lack Of)

Policy: no plaintext on black network

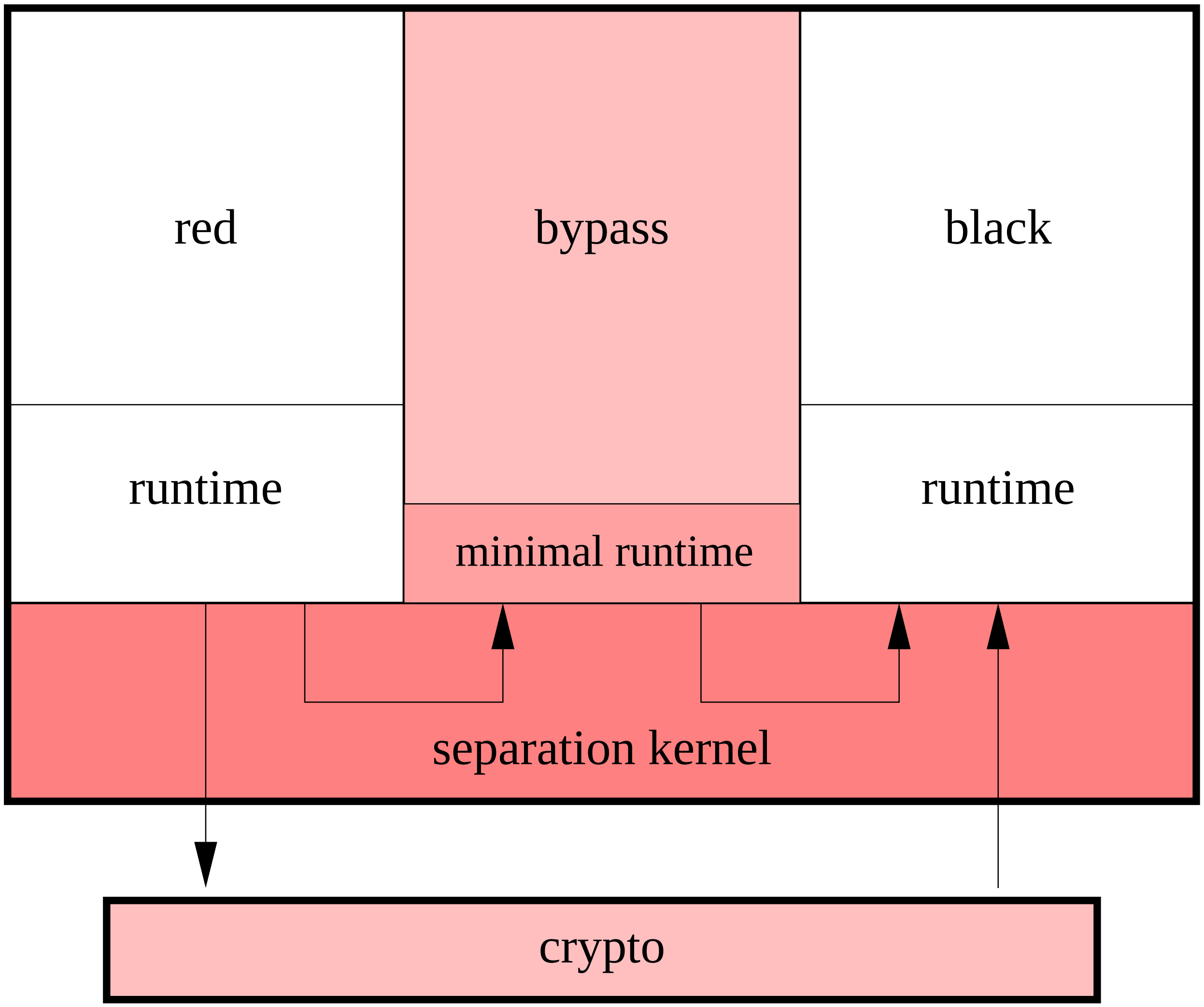


No architecture, everything trusted

Red-Black Separation



Red-Black Separation in the PFCs



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
- Maybe the approach used in the SUE and PFCs is preferable to the orthodox approach, at least for embedded systems and network components

Aha! 1981

Separate the issues of policy from those of resource sharing

1. Conceive of the system and its policy enforcement as a conceptually 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 minimal kernel to implement this conceptually distributed system in a single machine
   - Call that a separation kernel
   - All it does is separation, no policy

Design and Verification of Secure Systems, SOSP 1981

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)
- Brian's long-standing program in reliability and fault tolerance 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

Aha! 1982

- In 1981, we saw distributed systems as the conceptual model for secure architectures
- But the implementation was a logical simulation
    - Used 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
- But that would be wasteful for small components
- So you'd want a combination of logical and physical separation
- But there are further ways to realize separation
    - 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 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
- A Distributed Secure System, IEEE Computer 1983
- And A DSS: Then and Now ACSAC Classic Paper, 2007

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)
- Brian and I 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
- Naturally, Brian and I tend to attribute the problems to technical limitations of the time, pioneering use of middleware and distribution, and to UK development and management practices at the time
- 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)
- This is a reincarnation of DSS
- Used in F22, F35, FCS, JTRS, DDG-1000

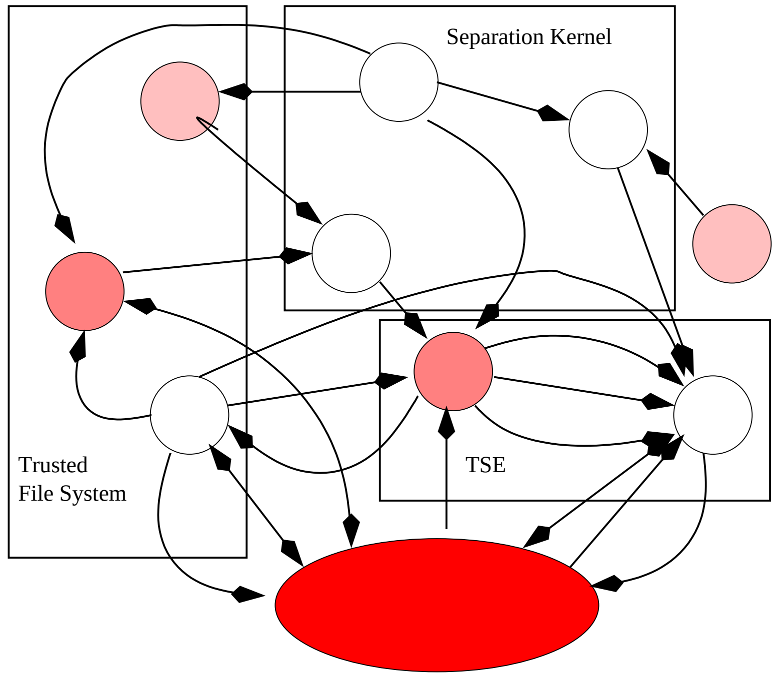
The MILS Version of DSS

- Conceive of the system policy architecture as a circles and arrows diagram
- Try to arrange it so that security depends on only a few trusted components
- And those are trusted to do only relatively simple things
- We can afford to have lots of circles and arrows, and can use this to reduce and simplify the trusted components
  - Split big components up
  - Replicate components at each level
- A separate class of resource sharing components (e.g., separation kernels) implements the policy architecture on physical resources

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
- So MILS requires a design approach that is partly top-down, and partly bottom-up (to use existing components)
- In DSS, assurance was left as an exercise for the reader
- MILS provides a compositional approach to assurance

A DSS/MILS Architecture



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

Looking forward (DSS/MILS)

- It is generally accepted that it takes about 25 years for a research idea to find its way into practice, so
- Be early
- Be patient
- Be right

  “It is a great advantage for a system of philosophy to be substantially true” [George Santayana]

Other Security-Related Topics

- J.E. Dobson and B. Randell: Building Reliable Secure Computing Systems Out of Unreliable Insecure Components
    - Cf. much current work
- Voting systems
- Cheating in Online Games
- A Computer Scientist's Reactions to NPfIT
    - National Health Service's National Programme for Information Technology (NPfIT)

Unreasonable Effectiveness

- In security, as in the many other topics reviewed today, Brian's contributions have proved singularly prescient, effective, and fertile
- What is the source of this unreasonable effectiveness?
    - Unreasonable because it looks effortless
- I think it's skilled deployment of the system perspective
    - Systems are complex; parts and properties interact
    - Hence the importance of structure
    - And the necessity of ecumenical thinking about critical properties: Concepts and Taxonomy of Dependable and Secure Computing (with Laprie et al)
    - And an integrated view of fault tolerance/fault avoidance (cf. redundancy vs. proof)

Thank You, Brian

- The title of the previous slide came from Wigner
- To paraphrase another part of his text
- The miracle of the appropriateness of Brian's approach to the formulation of problems and solutions in Computer Science is a wonderful gift which we ~~neither understand nor deserve~~ . . . can all learn from, and strive to apply

  We should be grateful for it and hope that it will remain valid in future research and that it will extend . . . to wide branches of learning
- I am sure all of us are grateful and have learned much from Brian's example
- And I certainly hope that we and he will continue to extend his wisdom to wide branches of learning