PSOS Revisited

Peter G. Neumann and
Richard J. Feiertag

ACSAC 2003
Classic Papers Track

- -

A novel conceptual architecture with tagged nonforgeable typed capabilities and hierarchical abstraction

PSOS Revisited

- -

Peter G. Neumann
Principal Scientist
Computer Science Laboratory
SRI International
Menlo Park, CA 94025-3493
Neumann@CSL.sri.com
www.csl.sri.com/neumann
Tel 1-650-859-2375

PSOS Revisited

- -

Richard J. Feiertag
Cougaar Software, Inc.
833 North Shoreline Blvd
Mountain View CA 94043
rfeiertag@cougaarsoftware.com
1-650-969-6960 ext 102

Primary Reference

- -

Peter G. Neumann, Robert S. Boyer, Richard J. Feiertag, Karl N. Levitt, and Larry Robinson,
A Provably Secure Operating System: The System, Its Applications, and Proofs,
Computer Science Laboratory, SRI International, Menlo Park CA, May 1980, Report CSL-116

Earlier Reference

- -

R.J. Feiertag and P.G. Neumann, The Foundations of a Provably Secure Operating System (PSOS), *Proceedings of the National Computer Conference,* **1979, pages 329–334 (on Neumann Web site)**

PSOS Design (1973-80)

- -

- **Pervasive capability addressing, tagged and nonforgeable in hardware, nonbypassable**
- **Hierarachical abstraction with modular encapsulation, object oriented, strongly typed**
- **Arbitrarily extensible (see tables)**

PSOS Capabilities

- -

- **Only two operations create capabilities: create new one, or create restricted copy**
- **All objects are capability addressed, nonbypassably, typed.**
- **Capabilities are openly accessible unless hidden by some layer.**
- **Capabilities can be tagged as propagation limited, if desired.**

Implementability

- -

- **Many lower-layer ops (*) directly executable from above.**
- **MLS could be embedded in layer 0 or as an object type.**
- **Hardware could have been easily retrofitted in 1980.**
- **PSOS-like typing is used in Honeywell/SCC secure systems.**

Formal Methodology: HDM

- -

- **The Hierarchical Development Methodology with SPECIAL (SPECIfication and Assertion Language) facilitated formal specification of modules interlayer state mappings, and abstract implementations, enabling full-system analysis**

Formal Methodology: HDM

- -

- **A complex design became conceptually simple through its hierarchy and composable abstraction and encapsulation.**
- **Each module spec around 2 pages, state mappings and abstract implementations straightforward and explicit, including pre-/post-conditions, exceptions.**

Lessons Learned

- -

- **Hierarchical design with modular encapsulation is a powerful design aid. It need not undermine efficiency, especially with appropriate hardware.**
- **A formal methodology can be very valuable for high-assurance critical functionality, especially security.**

More Lessons Learned

- -

- **Computer development is mostly an incremental process, driven by marketplace forces. But security research is slow to be adopted.**
- **There is no substitute for sound architectures and principled developments.**

Historical Perspectives

- -

Butler Lampson in the 1970s:

- **Capability systems are the way of the future, and always will be.**

(The same has also been said of formal methods.)

Contrarian view from the 2000s:

- **Shortsightedness is a way of the future, and always will be.**

URLs

- -

- **www.CSL.sri.com/neumann**
- **1979 Feiertag-Neumann paper:**

www.csl.sri.com/neumann/psos.pdf

- **Principled Assuredly Trustworthy Composable Architectures:**

www.CSL.sri.com/neumann/chats4.html, .pdf, .ps

- **RISKS archives: www.risks.org**

Minibio for Peter G. Neumann

- -

Peter G. Neumann (Neumann@CSL.sri.com) has doctorates from Harvard and Darmstadt. After 10 years at Bell Labs in Murray Hill, New Jersey, in the 1960s, he has been in SRI's Computer Science Lab since Sept 1971. He is concerned with computer systems and networks, security, reliability, survivability, safety, and many risks-related issues such as voting-system integrity, crypto policy, social implications, and human needs including privacy. He moderates the ACM Risks Forum, edits CACM's monthly Inside Risks column, chairs the ACM Committee on Computers and Public Policy, co-chairs the ACM Advisory Committee on Security and Privacy.

He co-founded People For Internet Responsibility (PFIR, http://www.PFIR.org). His book, Computer-Related Risks, is in its fifth printing. He is a Fellow of the ACM, IEEE, and AAAS, and is also an SRI Fellow. He is a member of the U.S. General Accounting Office Executive Council on Information Management and Technology, and the California Office of Privacy Protection Advisory Council. He has taught at Stanford, U.C. Berkeley, and the University of Maryland. For further background, see his Web site (http://www.csl.sri.com/neumann).

layer	PSOS Abstraction or Function
17+	applications and user code (-)
16	user request interpreter *
15	user environments and name spaces *
14	user input-output *
13	procedure records *
12	user processes*, visible input-output*
11	creation and deletion of user objects*
10	directories (*)[c11]
9	extended types (*)[c11]
8	segmentation (*)[c11]
7	paging [8]
6	system processes, input-output [12]
5	primitive input/output [6]
4	arithmetic, other basic operations *
3	clocks [6]
2	interrupts [6]
1	registers (*), addressable memory [7]
0	capabilities *
Note:	
*	user-visible interface
(*)	partially visible interface
(-)	user-restrictable as desired
[c11]	creation/deletion hidden by layer 11
[i]	module hidden by layer i=6,7,8, or 12

Group	PSOS Abstraction	layers
G	user/application activities	17–...
F	user abstractions	14–16
E	community abstractions	10–13
D	abstract object manager	9
C	virtual resources (HW/SW)	6–8
B	physical resources (HW)	1–5
A	capabilities (HW)	0

layer	Property
17+	Application-relevant properties
16	Soundness of user types
15	Search-path flaw avoidance
12	Process isolation, no residues,
11	No lost objects
9	Generic type soundness
8	Segmentation integrity (HW)
6	Interrupts properly masked (HW)
4	Correctness of basic operations (HW)
0	Nonforgeable, nonbypassable, nonalterable capabilities (HW)