

Hierarchies, Lowerarchies, Anarchies, Plutarchies: Historical Perspectives of Trustworthy Architectures

Peter G. Neumann,
Principal Scientist
SRI International ComputerSciLab
Menlo Park, CA 94025-3493
Neumann@CSL.sri.com 1-650-859-237
<http://www.csl.sri.com/neumann>
AFRL/CPSG Layered Assurance
San Antonio TX, 5 August 2009

1

Trustworthiness

Trustworthiness is a measure of the extent to which a system, network, person, or other entity is worthy of being trusted to satisfy desired requirements (e.g., for security, reliability, human safety, system survivability despite realistic adversities, interoperability, ...), under certain assumptions.

2

Untrustworthiness

Untrustworthiness often results, due to incomplete requirements (specified vs desired), unsound architectures, bad implementation, dependence on untrustworthy 3rd parties, insider misuse, human frailty, invalid assumptions, sloppy evaluations, faulty risk analysis, etc.

3

Trust vs Trustworthiness

Trust is a slippery slope.
You may trust some entity because:

- You believe it is trustworthy.
- You don't even know it exists.
- You are naïve or gullible.
- You ignore dialog boxes.
- You don't check certificates.
- You know no alternatives.

4

Trustworthy Layered Architectures

- Multics 1965, PSOS, SIFT 1973
- Rushby isolation kernel 1982, Rushby-DeLong 2007-2009
- MLS: KSOS, KVM, SeaView ...
- MILS: Rushby-Randell DSS 1983, NSA 1988, SRI 1992
- Virtualization: KVM, Rushby-DeLong 2007-2009
- Composability, e.g., www.csl.sri.com/neumann/chats4.pdf

5

Layered Trustworthiness Compromises

Compromise results from misuse, HW faults, system failures, acts of God, ..., with three basic types

- Compromise from above/outside: penetrations, denials of service, untrustworthy 3rd parties, clouds, human frailty
- Compromise from within: insiders
- Compromise from below: subversion (overt or otherwise)

6

More Architectural Issues

- Trustworthy critical components
- Sound bases for composition
- Trusted bootload, trusted paths
- Cryptographic authentication
- Finer-grained authorization
- Traceback abilities
- Trustworthy code distribution
- Don't forget denials of service
- Alternative hardware bases

7

Multics Hardware, 1965

- Virtual memory segments/pages, process isolation, address spaces, argument validation
- Access control interpretation
- Descriptor cache for performance
- Rings generalize supervisor/user to linearly-ordered domains, limit compromise from above.

8

Original Multics Software, 1965-1970

- Modular encapsulation, reentrant PL/I code; per-user processes; dynamically linked symbolic file & names with access controls, I/O; unified design philosophy – e.g., dependence on symbolic addressing, dynamic segment linkage, paging, stream I/O, command standards, conventions, canonicalization, ...

9

MLS Multics: AIM, 1972 (Access Isolation Mechanism)

Ring 1 MLS AIM provided 8 levels & 18 categories with very little performance degradation as Standard Multics feature (but seldom used). (8 MLS levels unrelated to 8 rings.) Secure audit logged failed accesses.

10

PSOS Design (SRI-NSA, 1973-80)

- Pervasive capability addressing, tagged, typed, nonforgeable in HW/SW, nonbypassable
- Hierarchical encapsulated modular abstraction, object-oriented typing, extensible, formally specified in SRI's Hierarchical Development Methodology (HDM)

11

PSOS Capabilities

- Only two operations create capabilities: create new one, or create restricted copy.
- All objects are capability addressed, nonbypassably.
- Incremental trustworthiness. Capabilities accessible unless hidden by some layer.
- Capabilities could be tagged as propagation limited, MLS/MLI

12

```

Layer  PSOS Abstraction or Functions
-----
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 * [MLS?]
-----
* 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

```

13

PSOS Implementability

- Many lower-layer ops * would be directly executable from above, although some were hidden [].
- Multilevel security (MLS) could be embedded in layer 0 or as a secure object type.
- Hardware easily retrofittable.
- PSOS-like typing is used in Honeywell/SCC secure systems LoCK, SAT, SideWinder

15

PSOS Principled Assurance

- Pervasive assurance throughout cycles of development and use
- Assured composability, layered hierarchical noncompromisibility (see Robinson-Levitt 1977). Cf. the CLInc stack.
- Assured multilevel security, with several possible alternative implementations.

14

SIFT (SRI/NASA)

- SIFT: Software Implemented Fault Tolerance, 1973-2000?:
Application layer
Voting layer (2 of 3)
Broadcast layer
Synchronization layer
7 Bendix avionics processors
Prototype ran nonstop at NASA Langley for many years.

16

SeaView (SRI/ONR-AFRL)

- Sea View, 1980s
 - Application layer
 - MLS-untrusted Oracle
 - MLS kernel(Composition: ‘balanced assurance’)

17

Formal Methodology: HDM/PVS

- Use of HDM facilitated formal specification of modules, interlayer state mappings, and abstract implementations, enabling bottom-to-top analysis.
- A complex design became conceptually simple through its hierarchy and composable abstraction and encapsulation.
- Cf. PVS ‘interpretations’

18

KSOS (Ford Aerospace)

34-function MLS kernel, formally specified in HDM; Feiertag MLS flow analyzer used Boyer-Moore to find security flaws in specifications, most of which were fixed (except for a few detected covert storage channels). Also, code-to-spec consistency proof feasibility.

19

KVM Layers (SDC)

MLS retrofit of kernel into IBM 370; NonKernel Control Program (one per security level); untrusted unmodified virtual operating system instances of MVS/MVT and VM/370 at desired MLS levels; users. Formal top- and 2nd-level specs formally verified with mappings in InaJo/FDM. Covert channel analysis.

20

KVM References

M. Schaefer, BD Gold, RR Linde, JF Scheid, Program confinement in KVM/370, Proc. 1977 ACM Annual Conference, Seattle, 404-410). (Flaw discovered in Amdahl HW!)

BD Gold, RR Linde, PF Cudney, KVM/370 in Retrospect, IEEE SSP, Oakland, 1984, 13–23.

21

Holistic System Approaches

- We need principled trustworthy systems with sound requirements, structured architectures, and proactive design for usability, evolvability, pervasive assurance; selective use of formal methods.

22

What Is Needed More Generally?

- Holistic analysis must address sustainable democracy, social equality, and environments; also proactive attention to aging infrastructures such as bridges, roads, levees, railroads, flood control, healthcare; perhaps most important: holistic education.

23

Table A-1. TABLE OF INTERDEPENDENCIES
[Source: Roadmap for Cybersecurity Research, 2009]

H,M,L (H=High, M=Medium, L=Low) are suggestive of the extent to which
 * X can contribute to the success of Y.
 * Y can benefit from progress in X.
 * Y may in some way depend on the trustworthiness of X.

		S	M	E	I	M	G	S	S	P	P	U	ROW TOTALS		
		C	E	V	N	A	I	U	I	R	R	S			
		A	T	A	S	L	D	R	T	O	I	A			
		L	R	L	I	W	M	V	U	V	V	B			
X: Topic	\ Y:	1	2	3	4	5	6	7	8	9	10	11	H	M	L
1 Scalable Trustwor	.	H	H	H	H	H	H	H	H	H	H	H	10	0	0
2 Enterpr Metrics	M	.	H	H	H	H	H	H	H	H	H	H	9	1	0
3 Eval Methodology	H	M	.	H	H	H	H	H	H	M	H	H	8	2	0
4 Coping w Insiders	H	M	M	.	H	M	M	H	M	M	H	H	4	6	0
5 Coping w Malware	H	M	M	M	.	M	H	H	M	M	H	H	4	6	0
6 Global ID Mgt	H	M	M	H	H	.	M	H	H	H	H	H	7	3	0
7 Sys Survivability	H	M	M	H	M	M	.	M	M	L	H	H	3	6	1
8 Situ-Attribution	M	M	M	H	H	M	H	.	M	M	H	H	4	6	0
9 Provenance	M	M	M	M	H	M	M	H	.	H	H	H	4	6	0
10 Privacy-Aware Sec	M	M	L	H	L	H	M	H	M	.	H	H	4	4	2
11 Usable Security	M	M	M	M	M	M	M	M	M	M	M	.	0	10	0
COLUMN TOTALS	H	5	1	2	7	7	4	5	8	4	4	9	57		
	M	5	9	7	3	2	6	5	2	6	5	1		50	
	L	0	0	1	0	1	0	0	0	0	1	0			3

24

Plutarchies

Plutarch's Greek writings stimulated among Romans considerable sense of the importance of understanding historical people and events. He observed that little seemed to have changed in human nature. Similarly, little has changed in commercial high-assurance systems, despite some major research advances. We need a better sense of history.

25

Saltzer-Schroeder-Kaashoek (1975+)

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design
- Minimization of secrets *
- Separation of privilege
- Least privilege
- Least common mechanism
- Psychological acceptability
- Work factor
- Monitoring/auditing

26

Saltzer-Schroeder-Kaashoek

- J.H. Saltzer & M.D. Schroeder
The Protection of Information in Computer Systems, *Proc. IEEE* 63, 9, September 1975, 1278–1308.
<http://www.multicians.org>
- J.H. Saltzer & F. Kaashoek,
Principles of Computer System Design, Morgan Kaufman, 2009,
Chapters 1-6; 7-11 are online.
<http://ocw.mit.edu/Saltzer-Kaashoek>

27

Some Computer-Related Conclusions

- We need proactive attention for computer-based infrastructures: trustworthy architectures, transparency, accountability...
- 20-20 foresight is much better than 20-20 hindsight. Priorities must be realistic and far-sighted.

28

Lessons Still To Be Learned

- Reliance on misapplied technology usually increases risks.
- With appropriate HW and system architecture, layered designs need not be inefficient.
- Eternal vigilance is required. (John Dewey: Each generation has to learn the lessons of the past all over again.)

29

Technological Desires

- Better system architectures
- Better system engineering
- Better public-private cooperation
- Better technology in education
- Practical privacy-aware crypto
- Nonproprietary systems: open source/arch/doc/composability
- (Leads to permanent job security?)

30

Possible Forcing Functions?

- Market forces are inadequate.
- Incentives for open systems, open interfaces, open source?
- Stronger regulation & liability?
- Tax incentives?
- Better awareness of the risks of untrustworthiness; disasters?
- Maybe some or all of the above?
- But there are no easy answers.

31

Overarching Lessons

- Don't overendow technology.
- Every would-be technological solution has some risks, and escalates the attacks.
- Attackers have many advantages over defenders. However, too often systems collapse on their own without provocation.

32

PSOS 1980 Report

- P.G. Neumann, R.S. Boyer, R.J. Feiertag, K.N. Levitt, L. Robinson, A Provably Secure Operating System: The System, Its Applications, and Proofs, SRI International, Computer Science Laboratory, 2nd edition, Report CSL-116, May 1980.
<http://www.csl.sri.com/neumann/psos/psos80.pdf> and [.ps](#)

33

1979 PSOS Paper

- R.J. Feiertag, P.G. Neumann, The Foundations of a Provably Secure Operating System (PSOS), Proceedings of the National Computer Conference, AFIPS Press, 1979, 329–334.
<http://www.csl.sri.com/neumann/psos.pdf>

34

2003 PSOS Revisited

- P.G. Neumann, R.J. Feiertag, PSOS Revisited, Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003), Classic Papers section, IEEE Computer Society, Las Vegas NV, December 2003, 208–216.
<http://www.csl.sri.com/neumann/psos03.pdf>

35

PGN References

- Reflections on System Trustworthiness, Advances in Computing v.70, 2007:
- Holistic Systems, *ACM SIGSOFT Softw.Eng.Notes*, Nov. 2006
<http://www.csl.sri.com/neumann/holistic.pdf>
- Principled Assuredly Trustworthy Composable Architectures, 2004:
<http://www.CSL.sri.com/neumann/chats4.html>, [.pdf](#), [.ps](#)

36

A Few More PGN References

- The Role of Motherhood in the Pop Art of System Programming, ACM 2nd SOSP, 1969: multicians.org/pgn-motherhood.htm
- Proctor-Neumann 1992, www.csl.sri.com/neumann/ncs92.html
- Computer-Related Risks, Addison-Wesley, 1995
- www.CSL.sri.com/neumann
- ACM Risks Forum, www.risks.org

37

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	6–8
B	physical resources	1–5
A	capabilities	0

39

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

38

layer	PSOS Abstraction or Function
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 *
	Ideally all in hardware.

40

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]
4	arithmetic, other basic operations *
1	registers (*)
0	capabilities *

layer	Properties for Analysis
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
6	Interrupts properly masked
4	Correctness of basic operations
0	Nonforgeable, nonbypassable, nonalterable capabilities MLS (if present in layer 0)