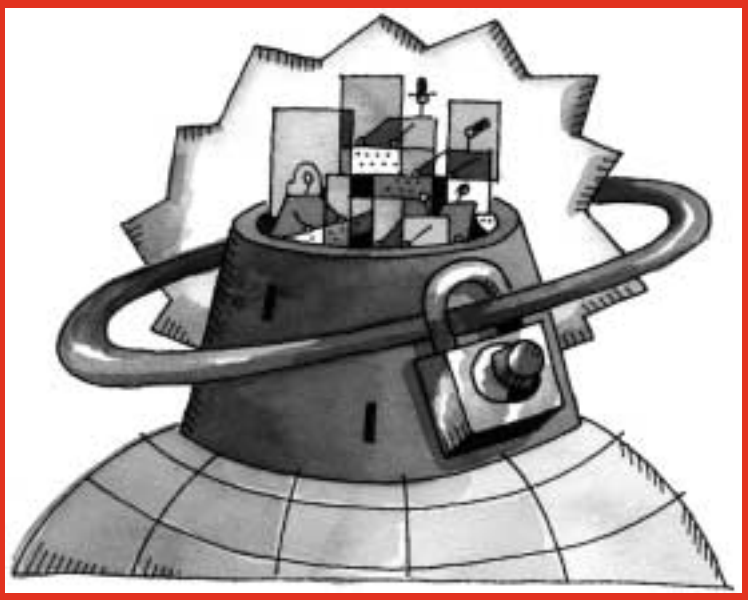
;login:
THE MAGAZINE OF USENIX & SAGE
December 2002 • volume 27 • number 6

Focus Issue: Security

Guest Editor: Rik Farrow

inside:



USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

the kernelized secure operating system (KSOS)

Last August I had the pleasure of attending the USENIX Security Symposium in San Francisco. As a security practitioner and former OS designer, I've always considered this one of my favorite conferences, and I haven't missed one in quite a while. But as I attended the panel discussion of Microsoft's Palladium and the TCPA, I was once again struck with a curious sense of déjà vu. Palladium in particular seems to be an attempt to create a "trusted" hardware system to overcome the inability to create large monolithic operating systems with acceptable software quality.

by Tom Perrine

Tom Perrine is the Manager of Security Technologies at the San Diego Supercomputer Center, where his job description is "protect the privacy and intellectual property of our users." Involved with computer security since the '80s, he was a Multician, has testified to Congress concerning Carnivore, and studies computer security technology and how it relates to people and public policy.

tep@ARPA.NET

It seemed to me that both Palladium and TCPA were rediscovering older security solutions to solve some of the same old problems. Much as I felt when Microsoft (and some of the open source OSes) announced the "invention" of symmetrical multiprocessing (SMP), I felt that someone, somewhere, had not done their homework. I felt the same way about VAX/VMS (and later, Windows NT) access control lists (ACLs), which were pale imitations of the Multics[1] ACLs.

There were SMP mainframes at least as far back as the 1960s in General Electric's GECOS[2] and, later, Multics operating systems – up to six processors in some cases. For those of you who don't remember, when Bell Labs pulled out of the GE/MIT Multics project (Project MAC), UNIX was conceived in part as a smaller, more practical implementation of the "Multics vision." IBM and (I think) UNIVAC also sold multiprocessor mainframes in the same era.

Now, I hate the "when I was a kid we had to feed wood into the boiler to power our steam computers" stories as much as everyone else. But, to be honest, the computer science and engineering disciplines do a terrible job of teaching the history of our field. And this leads to someone rediscovering the same old "new and novel" solutions to the same old problems about every five years, solutions that can often be found in the older literature.

Palladium and TCPA (and Linux and *BSD) continue to rediscover solutions that have been found before, in systems such as KVM/370,[3] Multics, SCOMP, and KSOS.

To understand KSOS and its place, you need to understand the time and background of its creation. It was the late 1970s and TOPS-10, TWENEX (later TOPS-20), UNIVAC EXEC 8, OS/360, and Multics were the denizens of this new thing called the ARPANET. The ARPANET backbone ran at 56Kbps and connected about 100 computers. There was also this new upstart thing that people were beginning to play with called UNIX, "6th Edition." It ran on these new 16-bit mini-computers from Digital Equipment, PDP-11s. Big ones, with 512K of core or semiconductor memory and 20MB washing machine drives. Programs were limited by the hardware to 64K bytes total or 64K instructions and 64K data, using split instruction and data addresses. SPLIT I/D was ugly. Don't ask. If you were there, I'm sorry to have reminded you.

There was a lot of interest in computer security. The government needed to process classified information, and computers were still expensive enough that they needed to be shared. There was lots of interest in allowing data at different classifications to be processed on the same computer at the same time, without "spilling" data across security levels.

1. The best current online information about Multics (Multiplexed Information and Computing Service) can be found at *http://www.multicians.org/*.

2. The following online entry has most of the story right. The main mistake is the claim that Multics had no database and no transaction processing. Other than that, it's pretty on target: *http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?GCOS*.

3. "KVM/370 in Retrospect," 1984 IEEE Symposium on Security and Privacy: *http://www.computer.org/proceedings/sp/0532/05320013abs.htm*.

The "Orange Book" hadn't been written yet. The experiences from developing Multics, SCOMP, and KSOS substantially influenced[4] the content of the Orange Book.[5] "Hackers" were still people who were curious about computers and wrote interesting code, mostly at MIT and Stanford AI Lab.

A few years earlier, in 1973, the seminal report on computer security – the "Anderson Report"[6] – had been published for the US Air Force. This report called for better software design practices, better programming languages, and something new called a "security kernel." It also suggested using formal mathematical models to prove that the kernel would operate correctly. This paper also, almost as an afterthought, described what we now call "automated intrusion detection" and noted that a primary way to compromise an operating system was to exploit "insufficient argument validation."

Yes, the Anderson Report described buffer overflows as a proven penetration method *and ways to avoid them* 30 years ago. We've obviously come a long way since then. So far that we need Palladium and TCPA.

The Anderson Report also cited an obscure little paper: "Notes on Structured Programming" by Edsger Dijkstra. This was about the same time that he wrote a letter to the *Journal of the ACM*, "Goto considered harmful." Strangely enough, it was the day of the Palladium/TCPA panel that we learned of his death. His two papers jump-started the entire structured programming movement of the 1970s and 1980s.

At about the same time (1973), a DoD-inspired mandatory formal security model was developed by Bell and LaPadula working at Mitre. This model formalized the DoD classification system into a set of mathematical rules called "mandatory access controls." The idea was that the site policy would override any "discretionary access controls" – that is, people could not give away data to unauthorized people. One rule ("simple security") prohibited data at a "higher" level from being read by a process at a "lower" level. Another rule ("*-property") prohibited a process at a "higher" level from writing to "lower" level data.

People started to design and develop "security kernels." These were small, well-defined cores upon which an OS could be written that would be small and "verifiable" using formal methods. This gets around the problem that verification methods and human minds weren't ready to deal with analyzing very complex systems. The idea was to concentrate all the security features, and only the security features, into a small kernel that would provide the base upon which a secure OS could be layered. This was imposing "least privilege" on the operating system itself, allowing the operating system to have bugs and yet not be able to compromise security. In 1976, Peter Neumann and others at SRI proposed "pSOS," a provably secure operating system.

In 1978, Neumann, John Nagle, and others at Ford Aerospace started work on a more ambitious project, which was actually expected to produce a running, practical, usable system, the Kernelized Secure Operating System (KSOS). Neumann went on to become the editor of *Risks Digest*, among many other projects. Nagle later worked in networking, producing Nagle's algorithm for merging tiny packets for TCP performance, which was published in RFC 896 and is part of every modern TCP/IP stack.

KSOS was intended to be a security kernel upon which a UNIX-like operating system could be built. It was recognized that the V6 UNIX kernel, at 10,000 lines of code[7] and 48(!) system calls, was much too big(!) to be formally specified or verified.

4. Jeff Makey, private communication. Jeff was an editor of the original Orange Book while at the National Computer Security Center.

5. Department of Defense, Trusted Computer System Evaluation Criteria: *http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html*.

6. J.P. Anderson, Computer Security Technology Planning Study, ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford, MA (Oct. 1972) [NTIS AD-758 206]: *http://seclab.cs.ucdavis.edu/projects/history/seminal.html.*

7. The "infamous" "Lions book," an annotated display of the UNIX V6 kernel for PDP-11, shows about 9000 lines of code, in 44 source files, including the .h files. Again available in print (after being quashed by AT&T in the seventies): John Lions, *Lion's Commentary on the UNIX 6th Edition with Source Code* (San Jose, CA: Peer-to-Peer Communications), 1996, ISBN 1-57398-013-7.

But by building a smaller security kernel that would implement the security features, a UNIX-compatible layer could be built on top of that "micro-kernel" that would provide a UNIX-compatible system-call interface. The micro-kernel approach lived on in later OSes such as MACH.

It was amusing to me that during the Palladium/TCPA panel, when asked about improving the quality of the operating system so that Palladium would be less necessary, the Microsoft speaker scoffed at "fixing millions of lines of code," implying that the OS was just too big to be written properly. Perhaps they need to "invent" least privilege and "micro-kernels" again!

But back to KSOS. First, the security kernel itself was designed and specified. The kernel was modeled as a finite-state machine, and the system calls were defined in terms of the state transitions that could occur. It was decided that by defining an initial known secure state, and then checking all possible state transitions (system calls) to make sure that they led to a known secure state, the kernel would be "verified." The design was documented and specified in the SPECIAL specification language. The kernel specification was considered manageable because there were only 32 kernel calls.

The kernel specification was examined using the Boyer-Moore theorem prover. The prover, which ran on a fairly large DEC 20, was eventually able, with *extensive* human assistance, to prove enough of the theorems to provide a significant level of assurance of "correctness." In this context, this means that the kernel specification contained no explicit violations of the Bell-LaPadula model, no "covert channels" or ways for data to be implicitly shared across mandatory access boundaries. Note that even with only 32 system calls, the specification had to be split into five pieces, which were separately verified. This verification allowed KSOS to be considered a candidate for an "A1" rating, "verified design," the highest Orange Book rating.

KSOS was implemented in a higher-level language, Modula-2. Think of the type safety of Pascal, with the package constructs of Ada or C++ (years before either language was designed). User programs could be written in Modula-2 or C. KSOS was not self-hosted; programs had to be compiled under V6 UNIX and copied to the KSOS system.

KSOS had some other interesting features for its time, such as multiple virtual terminals per real terminal (think of Linux virtual consoles), and a "trusted path" from the terminal into the security kernel. When a user hit the "secure attention" key (such as BREAK), all user programs were suspended and disconnected from the user's terminal. The terminal was connected straight to the kernel. This was to avoid applications from impersonating trusted applications or the kernel's authentication (login) or password change functions.

KSOS also implemented "shared memory segments," typed files, and what could be considered network firewall features – all before Berkeley completed 4.2BSD.

In 1981, KSOS development moved to Logicon in San Diego, where it was further enhanced and later served as the platform for several Navy and Air Force operational systems, such as ACCAT GUARD and USAFE GUARD. These systems used KSOS-hosted applications to provide multi-level secure application gateways on very secure DoD networks. The "final" version of KSOS for PDP-11 is described in Perrine, Codd, and Hardy,[8] which also includes some information about ACCAT GUARD.

8. Perrine, Codd, Hardy, "An Overview of the Kernelized Secure Operating System (KSOS)," Proceedings of the 7th DoD/NBS Computer Security Conference, September 24–26, 1984: *http://users.sdsc.edu/~tep/Presentations/Overview_Paper.text*.

Later, KSOS was ported to the VAX and became KSOS-32. That project was canceled, along with many other DoD computer security projects, in September 1988, shortly after the first user login was achieved.

Although KSOS (and SCOMP and Multics) made significant advances in computer security and software design methodologies and helped us to understand the problem of software quality and assurance, they have been mostly forgotten. These OSes, and their contemporaries, provided many features and services that are continually rediscovered or even "invented" every few years for new operating systems. Palladium and TCPA are just the most recent efforts to cover the same ground. In Orange Book terms, they are trying to go "beyond A1" into "trusted hardware," without first getting to B-level software architecture.

It may be that UNIX came along and swept up a new generation, and the "old skool" operating systems and their "old guard" were not able to pass along the accumulated knowledge. It may be that so many of the older papers and research and real-world experience are not available online and, hence, not findable with a quick Google search. Or it may be that the computer science and engineering curricula aren't covering the history of computing at all, let alone the history of secure computing. Whatever the reasons, we're losing a lot of time and energy rediscovering technology and re-visiting the same failed solutions over and over again.