An Interview with

MARVIN SCHAEFER

OH 435

Conducted by Jeffrey R. Yost

on

20 November 2013

Computer Security History Project

Columbia, Maryland

Marvin Schaefer Interview

20 November 2013

Oral History 435

Abstract

This interview with computer security pioneer Marvin Schaefer discusses his roles and perspectives on computer security work at the System Development Corporation over many years (an organization he began working at in the summer of 1965), as well as his work at the National Computer Security Center in helping to create the Trusted Computer System Evaluation Criteria (TCSEC). With the latter he relates the challenges to writing the criteria, the debates over the structure and levels, and the involvement of criteria lawyers. He also summarizes his work at the company Trusted Information Systems. In addition to detailing his pivotal work in computer security, he offers insightful commentary on issues in the field such as the Bell-LaPadula Model, John McLean's System Z, and other topics.

This material is based upon work supported by the National Science Foundation under Grant No. 1116862, "Building an Infrastructure for Computer Security History."

Yost: My name is Jeffrey Yost, from the University of Minnesota I'm here this afternoon, November 20, 2013 at the home of Marvin Schaefer in Woodbine, Maryland. This is an interview for CBI's NSF-sponsored project "Building an Infrastructure for Computer Security History." Marv, I'd like to begin just a couple biographical questions; can you tell me when and where you were born?

Schaefer: I was born in 1943 in Los Angeles, California; and moved when I was six to the San Fernando Valley, where I lived for more than half of my life. We moved to Fontenay-le-Fleury, France, for a bit less than a year. And then here to Maryland, thanks to Roger Schell coming to our home and convincing me I wanted to move from Southern California to the Computer Security Evaluation Center at NSA.

Yost: You went to school for undergrad at San Fernando Valley State College?

Schaefer: That's correct. And also for my master's degree.

Yost: And studied mathematics?

Schaefer: Yes.

Yost: Did you know from the start that you wanted to study mathematics, did you long have an interest in that area?

Schaefer: It was more fun than most things that I had encountered, but going through college I had a hard time deciding which major to actually complete. I was also interested in the French language, the English language, computational linguistics, music and philosophy; and it was fairly late when I finally decided which units to apply to a degree. I graduated as an undergraduate with 180 units in four years.

Yost: And you completed your master's degree there, as well?

Schaefer: In six months.

Yost: Through your undergraduate and master's degree, did you have any experience with using computers?

Schaefer: Only once. My wife and I both took a course using Daniel McCracken's book on digital programming, and I learned how to program in absolute binary. We never did learn a higher order language in that course, so it was strictly assembly language when we finally learned how to use an assembler, and it was absolute binary before that, primarily on an LGP-30 computer, though I did get some experience sending in batch work to run on the IBM 7090 at UCLA's Western Data Processing Center. In the final week of the semester, we were introduced to FORTRAN II, but did not have a chance to try running a program.

Yost: In 1967, you entered the Ph.D. program in mathematics at UCLA, is that correct?

Schaefer: Yes.

Yost: When did your first association with System Development Corporation start?

Schaefer: That was in the month of May, 1965. When I was looking for a summer job, I liked the advertisement that SDC had put into the *Notices of the American Mathematics Society*. Their logo was a rectangular paraboloid and it appealed to me, so I phoned them and found out they were starting a new research and technology directorate. They had a couple of openings left and if I'd come down and take an aptitude test, which turned out to be trivial, I could be placed in the class. And so we were exposed, in that class — I think there were 20 or so of us in the group — to the latest research at the time, which was largely based on the use of time-sharing; it included operating system theory, formal language theory, natural language and speech recognition, computer graphics and visualization, metacompilers, hierarchical database management, and some mathematical theorem proving by computer. System Development Corporation at that time was relatively newly formed, having been the former System Development Division of the RAND Corporation. And so there was close coupling, still, between the groups. SDC had just developed a time-sharing system. The only other time-sharing system was at M.I.T. at that time. SDC's was the Q-32 Time-Sharing System. They had an interpretive language called TINT, which was the Time-Shared INTerpreter of a dialect of JOVIAL. We were taught JOVIAL and, to some extent, TINT, by the people who developed JOVIAL originally for SAGE. Jules Schwartz was one of the lecturers; the acronym JOVIAL was

Jules' Own Version of the International Algorithmic Language [ALGOL]. We learned about ALGOL 60 from Tom Steel, who was on the ALGOL committee. We learned LISP, from Clark Weissman, who had implemented a version called LISP 1.5 on the Q-32 Time-Sharing System. We were exposed to a great deal of information about languages and compilers, natural language processing, graphics processing, some command and control, from people who were researching those areas. We learned about large scale database management, including a prototype of relational database management, as well as hierarchical database management from developers in those areas.

Yost: Were there any computer security design elements to Q-32?

Schaefer: No. It was a single-state computer. The closest we got to anything was a login process, but that was so that projects could be billed for the computer time they were using. The user's address space included the operating system's executing code as well as all of the user's code and data. There was also full, unencumbered access to all online disks and drums used as storage (and swapping) devices.

Yost: When you mentioned M.I.T., was that in reference to CTSS?

Schaefer: Yes. There was some rivalry at that time, between SDC/RAND and M.I.T. As I understood it at the time, both time-sharing systems came out at about the same time, within weeks of each other. The developers of that system, who included, to some extent

Clark Weissman, but Clay Fox, Martha Bleier, Jack Slaybaugh, and Salvador M. Aranda were the people who designed and coded the operating system. But as I say, it was a single state computer; it was a modification of the IBM-AN/FSQ-32. It was called the AN/FSQ-32V; the only major difference being that it had a timer in it for quanta, which were tunable. We were working with 300 milliseconds. It had four drums and a couple of disks, at the time. There was no paging system; every user space was basically swapped in entirely and moved out entirely for the next user as the machine time sliced. There was a total of 64k octal words on the machine; words were 48 bits long. The operating system took up all but 47K of that. So the largest possible environment for a user would be 47K, which, as it turned out, was not adequate for many of us who were doing research in either LISP or compiler-compilers, where we needed everything that was available. TINT itself, the Time-Sharing Interpreter was much smaller. I think it was around 16,000 so it wasn't too hard to get onto the machine if there weren't too many users at the same time.

Yost: Do you recall when the ADEPT-50 project started and were you at all involved?

Schaefer: That was either 1966 or 1967. ADEPT-50 actually came out in 1967, but I'm not sure if it actually started in 1966 or not. I wasn't directly involved in it. One day, Clark Weissman came to my office and told me about the security policy he was interested in putting on it, which was a high water mark model. He wanted to know its properties, and he had some specific questions with respect to user franchise, which was akin to access rights associated with the user's clearance. So he asked me to write a mathematical model and explore a few questions he posed to me. Writing the model didn't take very long; exploring the questions took a little bit longer but I think that most of that was concluded in a few hours, with more formalization as Clark would think of questions or I would think of aspects that I hadn't thought of before. So the whole process was done inside of a matter of a few weeks before he formalized it in his paper, which was developed to a paper at either a Spring or a Fall Joint Computer Conference.

Yost: There was a paper in 1967 and a prize winning one by him in 1969, "Security Controls in the ADEPT-50 Time-Sharing System"—Fall Joint Computer Conference, I believe.

Schaefer: I think it was 1967, and I think that may have been Atlantic City, but I'm not certain.

Yost: That same year, at the Spring Joint Computer Conference, Willis Ware and NSA's Bernard Peters gave a paper that was really the first public paper on multilevel security. Did you attend that and did that have any effect on you at the time?

Schaefer: I was at the conference and I remember prior discussions before the conference. So, Willis and R. Stockton Gaines, who we all called "Stock," had either come to our place or we'd gone over to his place to discuss some aspects of protection. And the main reasons that that came up at all were various younger people in the company — for some reason, all of them in the research or technology group — had started playing around with what they thought was the online payroll database, and were interested in penetrating it. It turned out that the people who were playing around with it, were seeking to play a trick. The normal trick was to change the salary of the president of the company to a dollar a year, after discovering what it was on the machine. The kids who were doing the changes didn't understand how accounting worked and so the normal checks and balances in the computer system always turned up the modifications, no one had to code anything in to check for the president's salary because some people were trying to give themselves raises. It turned out, internal inconsistencies always showed up. There was also an interesting guy in the company named Ellis Myer, whose first claim to fame ended up being only a sentence in the acknowledgements of RAND's 1955 book *A Million Random Digits with 100,000 Normal Deviates*. But Ellis had gone through to validate that all million digits were, in fact, random, according to the then-understanding of what randomness versus pseudo-randomness was. And so he spent months, essentially hand verifying the tables; then reading and writing certain programs to go through and do validations for normal number sequences. At SDC, Ellis was involved in the accounting and integrity checking code that was running on the Q-32 at the time. I think all of that ended up moving over to ADEPT-50 where younger people who had come into the company again tried their hand at modifying salaries. The salary database was always a prime target.

Yost: You mentioned there was a bit of a rivalry between SDC/RAND and M.I.T., contemporary to the ADEPT-50 project and the Multics project, a successor project to CTSS. Was there interaction and information exchange between the two?

Schaefer: I wasn't aware of it, if there was at that time. I didn't really know very much about Multics until maybe 1971, when Multics was getting enough press that we were discussing its features versus those of ADEPT-50, which was still being used. There was also a project that had started to come up with a different operating system for the System/360, and IBM was having trouble getting it reliable enough to actually work as a time-sharing system. OS/MVS or MVT, I can't remember which, was also being developed concurrent with that, and the systems were crashing frequently. ADEPT-50, at that point, was actually more reliable than the IBM-developed system and there was discussion of whether we should be looking at a model that was different from what IBM was doing, and Multics came into some discussions at that time. But I don't recall a firm rivalry at that point still pervading very much of what we were doing. Mind you, there was some competition for funding because one of the funding agencies, the Electronic Systems Division of the Air Force, ESD, was giving much of its money either to M.I.T. or to MITRE, whereas other aspects of the Air Force or other contracting agencies were giving funds to SDC and RAND. And so there was some competition for research dollars there. We were largely funded by ARPA, which didn't yet have a "D" in its name. In fact, from the time I joined SDC until I left, most of my funding came from ARPA.

Yost: Were you aware of the Anderson Committee's work as it was going on in the early 1970s?

Schaefer: Yes.

Yost: . . . and did that influence your thinking about computer security?

Schaefer: A little. Jim Anderson came by the company a number of times; he was a good friend of Clark's. We were still a very small division; I think we had maybe 30 or 40 people; and so when anyone from outside came in, we all had to know they were coming anyway, because of classified papers; but also, if they were interesting, we had some sort of get together. It wasn't a formal seminar by any means, but we certainly talked. Clark, having been very interested in security because of his relationship with the Ware Committee, had initiated a number of penetration studies of ADEPT-50 and much to his disgruntlement, every one of them succeeded in finding a new flaw in the system. And so various of us were asked if we could think of any reason for a difference in the protection mechanism that would help protect things. Now, this was done on an IBM 360, so there were two states, supervisor state and problem state, but it wasn't very hard to get to supervisor state on the machine. There was also discretionary access control and this high water mark problem, but for the most part, this was before the concept of a *process* versus a *program* was appreciated and formalized. Probably the biggest flaw in the modeling that I had done for Clark was the notion that we were looking at programs and didn't have the idea of system state that's reflected in the process. And so, one of the normal ways of subverting the system would be modifying the text of the program — which was necessarily unclassified or else one couldn't execute it — by logging in at the lowest possible level, which would be unclassified. There wasn't an integrity model built into the system, which is quite different from the way that Multics was developed, with its rings of protection. I don't recall Elliott Organick ever visiting us and so I don't know if there was much discussion at that time of protection rings at all.

Yost: When you mention 30 or 40 people at in the research and technology group at SDC, how many were focused on computer security by the early 1970s?

Schaefer: I would guess, by the time I left to move to France, there were only two or three who looked at it very hard for very long. When I had moved back, the Virtual Machine Facility (VM/370) was being studied in a joint effort with IBM Yorktown Heights and Poughkeepsie research groups.

Yost: So there was Clark Weissman, Clayton Fox…..?

Schaefer: Yes, it was Clark, Ray Phillips, Sal Aranda, and Clay Fox, primarily, who were looking at it. When I moved back, the VM/370 study was finishing. I got involved a little bit in that largely because I liked punning, and many of the attacks were pun-related attacks, where context would change the meaning of something in the system [e.g., simple data could be read into an area where a validated channel control program was stored prior to its being executed without additional "legality-checking"]. At the time, I don't recall whether it was Dick Linde or Ray Phillips who came up with the idea to check for such a time of use attack, but that ended up being one of the downfalls of VM/370. We were also, at the time, being exposed increasingly to UNIX, which was, of course, named— thanks to Peter Neumann— as a very stripped down version of Multics.

I was among many who discovered the vulnerabilities with buffer overflow, as a security problem that was intrinsic to many systems, but very much so in UNIX. I also found that the aging of pointers was then quite dangerous in UNIX, finally, recognizing that one could do anything one wanted to pointers in UNIX. Taking over a UNIX-based operating system was rather trivial in those days.

Yost: Moving back for just a minute, you mentioned the penetration studies. When did those first occur and was there any model with that, this was prior to the famed penetration studies of Multics closer to the mid-1970s?

Schaefer: There wasn't much of model of penetrating systems in 1970 or 1971 that I can recall. It was pretty much intuitive at the time. Dick Linde was beginning to think about ways of repeating penetrations without having to start all over. And so some of his penetration methodology was getting to the point where it was getting ready to be written up. But things were still pretty much, gee, we did it this way last time, can we do it this way again? And so it hadn't yet become a methodology for reliably taking down systems, although the folklore was certainly getting robust at the time.

Yost: In one article, you wrote about an almost romanticized activity with the Tiger Teams, can you expand on that?

Schaefer: I don't know that romanticized is quite the thing, but it was a lot of fun. It also had some negative aspects to it, and probably anecdotally, the most interesting one was when Clark had begun to look at penetration studies as sort of a business area. SDC was going through a transition at the time, from being a strict nonprofit to a not completely government restricted nonprofit. What we were doing to get contracts didn't have to be limited to the government but it had to still have some form of public benefit or good in it. We could not really advertise, but we could be approached by companies to do things for them as long as the company in question was doing something, again, that was considered part of the public good. At some point, a utility company that I can't identify for you (because of a nondisclosure agreement) was running one of the versions of an IBM operating system, and they had, as part of their accountability audit, wanted to have a security study done on them. And so various of us looked at the operating system, with Dick Linde leading the effort. The team wrote up a vulnerability report of serious gaping holes in the operating system. The utility company basically said, that's nonsense, no one would do *that*; we're not going to pay you; you pulled a hoax on us. Clark took umbrage and I think it was the only time I could recall of Clark getting as angry as he did. He decided that it was time for us to demonstrate the flaws, which was done remotely from Santa Monica. The takeover of the operating system was, as expected, complete and shortly after it was taken over, either Clark contacted them to let them know what was done and how, or they contacted Clark to find out what had happened. But in any case, the notion of a lawsuit had come up and the company felt, as many companies did as the time — reacted the same as a person who is either vandalized or the analogy at the time, a woman who had been raped — anger and fear were mixed together. They finally agreed that if no one involved in the study or working at SDC were ever to reveal the vulnerabilities or the name of the company, the company would pay us the amount of the original contract and pretty much made it clear it was a never-again proposition. Although a number of years later, after some fixes had been made by IBM in the operating system, the utility company asked for a repeat performance and were again upset by the results.

Yost: Was there any sense of what commercial needs would be for computer security in the financial or other industries, and was SDC in its transition to not be exclusively DoD, thinking about the finance industry and other industries and a range of customers and clients?

Schaefer: There were several aspects of that going on. One of them was the ill-fated Time-Shared Data Management System, TDMS, which also went under the name TS/DBMS [Time-Shared Database Management System]. It was built initially with ADEPT-50, ran terribly slowly, was a four-level inverted hierarchical DBMS. Although it had some relational aspects, it was very, very slow. And as a research prototype, it did well with what it did. But another part of the company wanted to move TDMS to market quickly, deciding to start advertising it. And one of the things they were advertising was the security that was built into it. Well, it wasn't really security so much built into it, as it was security built into the underlying operating system. But various organizations subscribed to TDMS, one of which was the Santa Monica Police Department, which, in a sense, came close to being a commercial interest because they had privacy concerns. They also had integrity concerns; it would be wrong for someone who had a criminal record to be able to get in and modify the database in advance of a trial, or to find out what evidence they had not disclosed. So there were security concerns there. There were some attempts to bring a bank on, (I can't remember which bank was involved); but then once the size of the database for the bank pushed ADEPT-50's addressing and storage limits, period, the system was in serious jeopardy. A tremendous amount of corporate resource was spent trying to fix things in the system, none of which really succeeded and the company lost its proverbial shirt in the process. I had been doing a lot of language research from the time that I started at SDC and early on I had become responsible for maintaining and extending a compiler-compiler, which was originally Meta 5 and became Meta 6, when I meddled enough with its original concept. An application that came to mind was an automatic translator of databases from one structure to another structure, based on its input forms, not on its internal structures. I developed a so-called user-oriented language called DBL, for database Language. DBL could be used as a compiler-compiler to translate arbitrary database syntax and semantic structures into those that were recognized by TDMS. [I had originally begun this work with another DBMS called LUCID.] This tool was used on a number of different applications, but as the banks consistently messed up the format of their databases with respect to what TDMS was structured to process. I was spending a fair amount of time working on this problem of translating from what was coded into what was intended. I had also, in the language research area, gotten into the area of global compiler optimization, or global program optimization, research that resulted in a textbook I published with Prentice Hall in 1972 (that had led to my moving to France). But back to the TDBMS story: SDC management brought together a small group of us, I think six in all, to analyze the entire structure of the TDBMS system, discover flaws in it, redesign its internals and optimize its performance. Our challenge was to make it operate 10 times faster than it was, and we got it up to almost 100 times faster than it had been running. It was a fun project, but still not sufficient to save TDBMS from its eventual downfall as a commercial product. But along the way, there were security problems that were starting to come up and those dealt with access, from both a security and an integrity perspective to various fields of data, which would be, in relational database terms, columns of data. The meat of the issues were the semantic relationships between those columns, and in some cases semantic relationships between different records within the overall database [i.e., satisfying the syntactic requirements for data was straightforward, but these requirements only guaranteed that specific data were numeric, fell with defined ranges, were names, or the like; what is missing is determining the values were semantically meaningful in a greater context – e.g., if in a record for an individual, the block for Registered Voter is 'yes', it should also be the case that the individual's record shows that the age, residency and citizenship requirements are consistently satisfied]. I think I was then the only one at SDC who was seriously investigating semantic integrity for databases, but that ended up being a theme that stayed with me because I recognized that there wasn't any significant difference between the system tables in an operating system and the tables in a relational database: one way to bring down an operating system was to introduce an inconsistency in the system's internal tables. And so, fairly early on, a nascent form of secure state (or correct state) was beginning to form with respect to internal system tables, this was nourished, certainly, by the TDMS study, but earlier work that had come up also proved to be relevant. Some of this work that had caused me to have to commute to D.C. once a week during the Vietnam War. The Pentagon was having trouble with internal consistency on the Status of Forces database for Vietnam. One of the troubles they had was the keypunch operators who were getting the data ready for input didn't understand how to use the tab key, or the skip key, or the format key for the drums on the keypunch machine. So data were often entered in the wrong columns, smushed together, spelt inconsistently, etc. I ended up writing various programs in the compiler-compiler to either discover and convert obviously erroneous database entries into semantically correct database entries, or to reject them because they were too garbled. That ended up going on for, I think, around nine months. I'd participate in a seminar at UCLA in the morning on Monday; hop the flight to D.C.; if I had work to do I would normally handwrite it on the plane, give it to an SDC secretary who would transcribe it and send it by pouch back to the UCLA professor who was running the seminar so that I could get feedback when I returned in the afternoon on Friday. That got to be a horrible schedule and it finally just caused me to stop because as it turned out, the conjecture I had made for my dissertation was proved by someone at Princeton, which meant I got to start over. That really killed my enthusiasm to start over.

Yost: Can you briefly describe your Petri net research?

Schaefer: Yes. I'd forgotten about that. It was back during the Watergate hearings. I remember it well. I had had to take care of my wife and I was working from home at the time, listening to Watergate, typing on a time shared 2731 terminal over a modem at the same time, to write our database security model with Tom Hinke. We needed to incorporate a synchronization mechanism into the model to permit simultaneous accesses and updates to a database that would be consistent with secure dataflow constraints. A synchronization problem came about because of the paper that got Clark Weissman and Edsger Dijkstra into a strong argument over the Dining Philosopher's Problem. Somewhere along that time, I guess it had to again do with some compiler work I was doing with Tom Cheatham, I met Anatol W. Holt and learned about Petri nets. Petri nets looked, though complicated, to be a much better way of doing database synchronization than the equivalent form, which was Dijkstra's semaphores [the latter could be used for leaking sensitive data to unclassified levels]. And so there was the question of how. I have to go in and say something about the constraints we had on the database model before I say anything about the Petri net research. Then Major Roger Schell had let a contract to us from ESD to come up with an implementable secure database model that could operate with Air Force Data Services Center Multics, under the constraint that its implementation could have no security relevant code in it that would reside in either ring zero or ring one of Multics. This was largely because the security model interpretation from Ken Biba and the Bell-LaPadula model being applied in such a way that neither ring zero or ring one could be modified in Multics. So we had to come up with a way of implementing multilevel security in database management system without violating any of the Multics rules or contractual constraints. This proved to be very, very difficult for us to envision until largely I came up with the model based heavily on the principle of least privilege that together we formalized. Tom's engineering pragmatism was essential to this effort and its practical refinement. There were actually two papers: the preliminary model, of which I was principal author, and the final one that included implementation considerations of which he was principal author. We bounced ideas off each other when we found ourselves trapped by the constraints of implementation and feasibility. The hardest question was how to lock a transaction without violating the information-flow rules such that integrity could be preserved and one wouldn't have to worry about concurrent operations from another user operating at a different security level. We knew what a process was; we'd finally graduated from the idea of programs to the Multics concept. The Petri net research showed how to do it, essentially: how to lock things such that one could not necessarily guarantee that the transaction was going to terminate, but that one could absolutely determine whether or not there'd been interference. Why was this hard? Well, if one was updating, one essentially had to start with the primary key, which was unclassified, and hence, visible, but if other fields in the database had higher classifications one had to worry about the fact that another user at a lower level could modify a field, even as one operating at a higher level, could be reading or altering a more classified field. We derived an information-flow-secure method of adding transaction numbers, so that when one was starting one's update, one would read and increment one number, and when one finished, one would increment another transaction number. The integrity check would require that the second number was equal to the first number. That was derived from the Petri net model and I think that was the most valuable thing that we contributed to concurrent or asynchronous processing in multilevel systems; and it was also used in at least one operating system design for a trusted system. Some of that idea originated with Leslie Lamport, who was working with Tally Holt then, and some synchronization papers that the two of them had been working on, part of it was an original contribution at my end. Holt and Leslie Lamport and I were in fairly close communication during that period. The integrity lock, as we called the synchronism technique, was probably the most valuable thing that came out of that relationship. The problem with the mechanism is that I couldn't be guaranteed that a higher classification level update would necessarily terminate favorably in the presence of lower classification level actions. You could tell if something could have gone wrong, but being at the higher level when one was making a modification, one could never tell the people at the lower level that they had done something that that could corrupt what one was doing at the higher level; this is a consequence of the *-Property, essentially. And so the most sensitive data on the database was exposed to a potential integrity vulnerability by the least trustworthy people on the operating system, or on the database management system. This consequence was never very satisfying. But as we had recognized, we preserved information security, even if we had trouble completing semi-atomic transactions and preserving integrity. The other part of that problem was that there were more lower cleared people than higher cleared people, so in a volatile database, the starvation of the higher level updates could be a serious problem just because the frequency where the lower data was being accessed.

Yost: In the 1970s and 1980s, what were the most important venues for sharing knowledge and information in the community? Did you regularly attend the National Computer Security Conference, NBS meetings?

Schaefer: There wasn't a National Computer Security Conference in the 1970s. That didn't come until around 1980. These came about through the NBS/DoD computer security initiative that Steve Walker at the Pentagon and Jim Burroughs at NBS had pulled together.

Yost: That National Computer Security Conference was launched at about the same time as the Security and Privacy Symposium started with IEEE? 1980/81.

Schaefer: That makes sense, but those two events were independent.

Yost: Prior to that, Steve Walker organized a series of meetings?

Schaefer: Steve Walker had started one other thing, which was a Computer Security Working Group, which Clark and I were both members of; Peter Neumann was a member of it; Stock Gaines was in it; Jerry Popek from UCLA was in it; Butler Lampson, of course, was in it; Les Lamport occasionally joined in; Steve Lipner and Ed Burke from MITRE, Dan Edwards, Anne Marmor-Squires, Ken Schotting, and Hilda Faust, [now Hilda Faust-Matthews] from NSA; Jerry Saltzer and Mike Schroeder from MIT; and we would get together a few times a year to discuss computer security-related issues, and to divulge information about new penetration techniques we were finding. In 1976, Clark Weissman asked me to direct a project to solve the VM/370 problem and we came up with a design for a kernelized VM/370. We established an arms-length discussion relationship with IBM, but there was considerable difficulty working with IBM because of commercial and proprietary interests regarding their product. We were to use a variant on the Bell-LaPadula model, we were to limit ourselves to the amount of supervisor state code that we would modify in the VM/370, and of necessity, try to minimize or eliminate completely anything we would do to the control program. So I assembled, at that time, a small team to do a feasibility study. Steve Walker at, I think it was still ARPA but it might've been DARPA by then, had also started two different security UNIX projects, one of which became KSOS, or the Kernelized Secure Operating System; the was just called Secure UNIX until Bell Labs got involved in not wanting to have UNIX defamed by having to put the word 'secure' in front of it. I forget what the second study ended up being called, but there were two studies which were heavily funded to get a secure UNIX. And then there was ours, which was very minimally funded, to come up with a solution to the VM/370. Up until then, VM was still thought to be pretty darn secure, despite the joint report that was published in the *IBM Systems Journal*. And IBM, of course, was trying its own things to try to secure their system against penetration, even as we were trying to produce a multilevel secure version. And so for the remainder of my tenure at SDC, I was leading, among other things, the project to come up with a secure version of VM. We had originally wanted to do the programming in the high order language Euclid, which would be verifiable by machine. There was another joint effort between us, the University of Toronto, and the USC Information Sciences Institute with Ralph London to come up with a language design that would be fully verifiable by computer. SDC had a contract with ARPA both to implement this language, Euclid, and to come with a verification tool so that computer programs could be verified in it. Difficulties started surfacing in coming up with a language that one could be used to code an efficient operating system and still verify. [This problem also showed up a decade later with the DoD Ada effort.] SDC finally threw up its hands on that and the language Euclid got modified considerably into a language called EADS at the University of Toronto. We ended up resorting to throwing out the idea of verifiability when our funding was lowered, and decided to write the KVM/370 kernel in assembly language and to some extent, in the language JOVIAL.

Yost: Can you expand upon a topic you mentioned about detecting, being among the first to detect buffer overflows in UNIX and how that research and understanding came about, who you collaborated with.

Schaefer: Almost accidentally. In fact, it was accidental. What happened was I didn’t understand the differences between Multics implementation and those in UNIX, and I had gotten really used to using Multics, or really, ARPANET to do my text editing in Emacs, and decided that I could do that same thing using whatever the counterpart to Emacs was on our in-house UNIX system. It might’ve been called “e” or something because UNIX seems to like one-letter names that aren’t memorable. And I figured through the formatting program, I could send an entire buffer or a document I was writing to the counterpart that would do the formatting. As a consequence, I was overflowing buffers and crashing the entire operating system, which led to the complaints from all the other SDC users who weren’t running into what I was doing to them. This caused me to think very seriously about the question of what was happening and what else was going to happen. It became quite clear it was a pointer problem. The pointer was always to a buffer. There was a problem with the size of the document I was writing and the number of bits that were allocated to the length of the buffer, which I exceeded, and that led to discovering a number of anomalies in buffer overflow. UNIX had a secure password system, which was supposed to solve the world’s problems and I’ve no idea how many other people independently discovered this, but I remember one day looking at the code, because I was curious about how the encryption worked. I saw that the place where the comparison was done in the operating system was not very far from the buffer that one was typing into, which was *before* the place where the comparison would be made of the encrypted password. I found with some experimentation that if I typed in a ridiculously long password when it would ask for the password, where the first part and the end part were encrypted images of one another, that it would successfully mess up what was in between the two comparison sites to allow a login. That was one of my proudest achievements that year. I found later that I wasn't the only one who had discovered it, but it certainly got me onto accounts when my account was taken away from me for having crashed the system too many times. I remember very well, upsetting my then-boss at SDC by sending and receiving e-mail from him when I had no legitimate access to the system. I think I resisted for two years telling him how I was doing it. I had mentioned it at one of the security group meetings and found out that I wasn't the only one who had discovered it, but it turned out there wasn't a good security mechanism in UNIX in those days. So when a flaw was detected, maybe an installation would correct their version of the system, but Bell Labs didn't necessarily either get word or make the modifications themselves. So I do know that that particular buffer overflow flaw came about because I had looked closely at source code, and I think the reason I had looked at the source was I had crashed the system enough times that my account was taken away. But it wasn't really taken away, the login ID was there; it's just that it had no password.

Yost: Can you tell me about your decision to leave SDC to go to the newly formed National Computer Security Center and begin the work for the evaluation criteria?

Schaefer: I'd been involved since the NBS conference workshop in Miami, a year or two years before, when we were trying to evolve various issues in security and integrity. I wasn't on Ted Lee's panel; I had participated prior to that in coming up with the bull's eye idea for evaluation criteria. I think at that time I'd also been involved in some of Steve Walker's small teams for going out and to see what various companies were trying to do with coming up with better operating systems. Those went on for quite a while, up to the day that I left SDC. At the NBS Miami workshop, I was involved in financial integrity issues, with a different panel. That was an interesting, but frustrating, panel – data theft was considered an irrelevant consideration by this group. Sheila Brand, and I think Bob Abbott, were on the panel with me, along with mainly people from banks and insurance companies, and a lot of auditors. But we were doing a fair amount of monkeying around with what would be meant by assurance for a secure system. I forgot what your original question was, leading to that direction, but Roger Schell, Clark Weissman, Steve Walker, Denny Branstad were all there. Jerry Popek was an active participant in the criteria panel. We talked a lot about the criteria concept. I was more interested in that than the security and integrity problems of databases because the auditors didn't really care about much other than having an audit trail, and they didn't think anyone would ever modify an audit trail. Serious mistake. Side issue there, if one penetrated a system, one didn't want to leave fingerprints and so part of a good penetration would be modifying the audit trail to just before one took over the operating system. As systems started putting time stamps on files, the first thing one would do is read the time stamps on the files one was going to play with and reset those also, prior to leaving the system. So from my point of view as a hard core security type, the assurances that the auditing community wanted just weren't real good assurances. Anyway, I had participated in a number of good workshops and discussions by the time that the Computer Security Center was being formed; and was involved in several of these proto-evaluation efforts with different vendors, under Steve Walker's auspices. The result was I was one of just a few people who had experience who was really interested in pursuing this matter. When Roger went to one of the Oakland Security and Privacy meetings he talked about what the center was going to be doing. Privately, we talked afterward and left that meeting with a kind of "we'll think about it." Not too long after that, Roger contacted me and wanted to know if I'd be willing to think about it more. Had a couple of exchanges with him; a couple of exchanges with George Cotter at NSA, who at the time was going to be running the Computer Security Evaluation Center. I was still having strong doubts. I really liked living in the house we had recently purchased in the west San Fernando Valley when Roger invited himself to stay over one night, just before Christmas and talk to us. And that pretty much turned it. It looked from his discussion of research coming into the center, and the outline he had of what else was going to be done by the Computer Security Center, that I decided to make the move. It was almost four months to get a security clearance through the system. I told SDC about it very early on and they were fully supportive, and so when the clearance finally came through and the glitch that was holding it up got identified and resolved, I drove across the country and reported the next Monday. The security clearance glitch [was that] on the form that one filled in to get the clearance, one of the questions was what is your current clearance? And I said I had a vanilla top secret clearance. The people in NSA’s security group that were doing the background work were having trouble finding out what program was associated with VANILLA, as a code word, and it was probably two months into the process when I finally got a secure phone call at SDC asking me to give them more information about VANILLA. That is when I finally told them no, no, no; there is no such program, ‘vanilla’ means it’s unadorned, it’s just straight industrial top secret clearance; and at that point, things picked up.

Yost: What were your expectations for the impact that the design of the criteria and the publication of TCSEC would have?

Schaefer: Well, we were having success working with industry, to start with. I was involved with three or four different groups, one of which was meeting regularly in Minneapolis/St. Paul — you can guess which company — but they were very energetic in what they were doing; they were quite enthusiastic. All of them were quite enthusiastic and it looked like it would be a good opportunity to move the field by having sponsorship and something official behind it to cause companies to want to make commercially viable operating systems that were secure. We recognized among ourselves, at least, that the lower divisions of criteria were harder to meet than the higher divisions because discretionary access control is not a solvable problem while labeled security genre is. So one emphasis was the ability to cause more companies to participate since they’d get an official imprimatur of the government if they succeeded. And the other was what was going to be happening, from my perspective, with research. I was strongly involved in formal verification activities and I was certainly interested in operating system security issues and database management security issues. I wasn't very much into network security issues at the time because that was a harder problem and I was happy looking at something that might be solvable. I didn't realize that the center was also going to become the DoD and defense community focal point in certification and accreditation of classified systems, which it became. There was also the educational role and the conference role of the organization, which were quite interesting to me. That was also the arm that would be doing the publication. So it was just a very, very enticing opportunity to get in and cause something to happen. In working for SDC there was always the issue that we were competing with one of the companies we were working with. In working for the government, that just went away and so I was quite happy to have that nonaffiliated imprimatur for that period.

Yost: Among the internal group, were there any major debates about the how the criteria would be set, how many levels and what they would be, can you recount that?

Schaefer: Yes. The first one was the question of whether we were going to try to do something with the bull's eyes, or Grace Nibaldi's criteria, or something else altogether, which would have been Chinese menu; one from column a, and one from column b, and so on. That position was strongly advocated by a group within MITRE. Ann Marie Discepolo, and MITRE had indeed drawn up a whole table of features and assurances that could be chosen from. Jim Anderson and Roger Schell and I didn't like that and the main reason was that it was too hard to figure out how to compare systems that way. Roger was very adamant that people who served as procurement officers had to have things in a simple form so that they could understand what they were doing and a rating of Level 3 or Level 4 was a lot easier to understand than one from column a and one from column b. So simplicity became very important. The next question that came up was Level 1, Level 2, Level 3, or something else? It was Dan Edwards who came up with the idea of rating the levels with a combination of a letter and a number, the way that bonds were rated by Standard & Poor's. What went into the levels was yet another issue, and that got debated for a long, long time with things moving up and moving down as requirements, as things evolved. The first version that we floated, which was distributed for comment as the Powder Blue criteria, came out with one extra level, A2, which was beyond A1, and it had a potpourri of features and assurances that were beyond the state of the art. Jim Anderson largely argued that "you can't require something that's beyond the state of the art," the state of the art will move, and therefore, the criteria will still be ahead of it, and therefore, it can never be satisfied. So we came up with A1, and nothing beyond A1, except something called "Beyond A1" with things that might be there. Discretionary access control and audit were very important and we didn't really think for a long time that there would be a reason to have C1. I forgot, per your note, what it was that disqualified RACF ultimately from getting C2, but we felt that, as Dan Edwards called it, "security with training wheels" was a good idea, and so C1 came about. The features in C1 and C2 were what we thought were largely state of the art for commercial operating systems that had some security in them. And we, after the Powder Blue Criteria, especially, got a lot of feedback on what industry thought of that, and the subsequent editions of different colors of the TCSEC reflected those comments from industry very strongly. B1 was literally C2 with labels, rather than C1 with labels, and it was thought to be the first step that could actually be achieved with minor modifications of a C2 system to satisfy some needs of the DoD but there were a lot of arguments at the time of what would go in. And there was also feedback from another part of the center that was beginning to get involved in commercial evaluations. The other part of the center that I mentioned was the part that was doing certifications and accreditations, and they were starting to try out the criteria on different systems, and find places where things just weren't fitting. So that ended up going in, along with the several hundreds of comments that came to us for what the criteria ultimately became. It was a report that was done by committee. Sheila Brand did a very good job of pulling in industrial comments and putting them into a form that we could work with. Sheila, I think, started with NBS and had a very good relationship with the commercial sector and that relationship helped us greatly. There was also still a prevailing view among much of industry that there was no reason to put security into operating systems because "no one would do that," in terms of breaking in and stealing stuff. And that attitude was present, not just in the commercial sector, but also within NSA itself. Everyone who worked in NSA had a top secret clearance, everyone had gone through a polygraph, and therefore, why protect anything? Everyone is cleared to see everything that's there anyway, and if you really wanted to get something there are lots of easier ways to get it than penetrating an operating system.

Yost: Was there any debate about whether the center would be at NSA, NBS, or [pause]

Schaefer: Yes, there was debate and Steve Walker can probably give you the best answer as to what happened there, and if not, Denny Branstad certainly could, from the other side. Part of the issue was, well, this is going to have to apply to classified systems also, and NBS really doesn't want to be involved in classified stuff. But the other was, the DoD would never accept anything from the Department of Commerce, and Steve got that very directly reported to him by numerous people within the Pentagon, where he has been working. So the hard issue they got Admiral Inman involved in before he left NSA, was the whole issue of the center being able to do unclassified work, and publish openly, and to hold conferences, which was completely antithetical to the way NSA worked at the time. To make things worse, when it was decided within the Pentagon, and with negotiations with Inman, that the Computer Security Center would be responsible for the consolidated R&D program and all of its funding. That also didn't go over well, not only with NSA or the other three-letter agencies, but also with the Department of the Air Force, Army, Marines, and Navy, all of which wanted to do their own research with their own budgets, and all of whom got very angry very quickly because all of those monies were moved to NSA. That added even more antipathy when the first consolidated program was attempted to be rolled out because NSA had never executed a computer security R&D program of that size with any credibility. So the comptroller in the Pentagon cut the budget to be more in line with what NSA used to spend, ignoring the fact that most of the execution would be by the Army, Navy, Air Force and their contractors. And so there was a good deal of computer security research and network security research that just didn't happen in the early years because of problems with the Pentagon comptroller's office. This idea of things being open and unclassified did not go over well with veteran NSAers, either, and there was a serious effort within NSA to kill the Computer Security Center before it got off the ground. That was led by the deputy directors for Research, and for Communication Security and the issue was constantly being placed on the desk of General Lincoln Faurer, who had become the Director of NSA. As this effort or that effort to kill the center or to kill its budget was consistently launched over and over again, the deputy director for Operations also got involved because there was a whole issue of the Center making our research publically known on how to secure systems through unclassified publications, seminars, conferences, and classroom settings. Since that time, well, I guess Mr. Snowden has taken care of the question of whether NSA had interests in seeing what was on other computer systems. It's now public knowledge that some parts of NSA were very much concerned with impediments to their ability to obtain access into foreign systems. And so there was aggressive argumentation (to be terribly euphemistic) over the existence of the Center and the details of its mission. Indeed, there were open, very bloody battles. Two days after I was in my office, which I then shared with Dan Edwards, I received a memorandum that I didn't think I even had a clearance to read. It was basically a challenge from the Operations side of the house to a paper that two people in the Center's research group were going to present at an EASCON conference on the benefits of using formal analysis of computer systems and computer models; basically, "use verification, it's good for you." There was no classified content in the paper other than what the Operations people were objecting to, to wit the very notion that if NSA was endorsing something, then foreign powers would a) know what NSA thought was important; and b) find ways of defeating it. And so I spent — that conference was to be the following Monday, this was

Wednesday — the next two days fighting, literally, in the office of the NSA chief scientist and what we were calling “the Ministry of Truth” (the organization that had already approved the paper for open publication and presentation). The author of the objection had raised the question of whether or not we were going to have to go over to the conference center that Sunday night and cut pages out of the printed and bound conference proceedings and cancel the presentation or not. Ultimately, Director Faurer decided that there was nothing in the paper that he considered objectionable, so we could go forward as planned, but in the future we would be censored and have to deal with both the Ministry of Truth and with the person who had written the very classified memorandum before we published anything else in public. That policy continued through the rest of my tenure at NSA.

Yost: Did that environment, both the conflict and uncertainty with the Center, as well as efforts at secrecy, have an impact on you deciding to leave and go to TIS?

Schaefer: Well, that was certainly part of it, but it was the general political system within NSA that ultimately got me to want to go back to doing things where I could actually get something done during my lifetime. Life at NSA was often a series of wall to wall meetings. The continuous challenges of one’s ability to do almost anything without having every office in the Agency comment and debate weighed far too heavily, and there was just too much wheel spinning in coordination. There was another problem that you had hinted at with one of your questions, and that was having contractual lawyers involved in writing the criteria. There were ambiguities we were not aware of as well as some that were intentional to permit creativity. These ambiguities were a serious issue to deal with, and we did not realize how serious a problem they would prove to be. When we wrote the TCSEC, we thought we were writing sufficiently clearly. And I remember when we tried to write some requirements as if they would fit in with contract monitoring activities, but they were still written by us as lay technologists, rather than as contractual lawyers. When positions in the Center became open and we were recruiting either new hires from universities or people from other parts of NSA to go through their career rotation to work with us, we inherited a number of people who were afraid to make a decision that hadn't been based on existing precedent. Much of their concern ended up being focused on the wording of specific requirements in the published criteria. As a result, a new profession we pejoratively called "criteria lawyer" came about in which almost anything a vendor tried to do was challenged because it didn't exactly meet a concept or requirement that was specified in the criteria. The impact of this was on what we thought would be fairly fast evaluations — like RACF, which after all was the "worked example" on which the C2 criteria were based to start with — turned out to be an impediment to getting a C2 evaluation completed. In fact, even getting a C1 evaluation through was hard because of specific wording in the criteria that proved to be ambiguous. And so that was one serious error in our writing of the TCSEC; almost none of the letters that Sheila received, addressed ambiguities in the wording. Evidently the people in the companies who were reading the draft criteria, read in the meanings that they thought made sense to them. We had written the drafts with meanings that made sense to us. We learned to our discomfiture that upon publication of the TCSEC, as far as the evaluators were concerned, we were all to be considered as if dead, so that the only meaning that could be ascribed to specific requirements were the wording that was in print. They could not go back and ask the dead authors what they meant [or thought they meant – for there were indeed differences in opinion that had not become apparent to us].

Mrs. Schaefer: [Laughing.] I think that's pitiful.

Schaefer: It was pitiful. And so Roger Schell, who unfortunately didn't remain in the Center very long, and Steve Lipner (who was not directly involved but ***was*** directly involved); and Clark Weissman (similarly); all found that they were no longer living citizens of the evaluation community.

Yost: You were chief scientist and vice president at Trusted Information Systems from 1986 to 1992, can you tell me about the principal things you worked on during those half dozen years?

Schaefer: I guess the first activity I got involved in was a system that was under development for the A1 level, or at worst the B3 level, by a company in Georgia. Steve Walker had gotten the contract, so Steve, Marty Branstad, David Bell, Pam Cochran and I were the consultants to that company's research effort. It was largely a capability-based operating system. We were very much aware of Butler Lampson's celebrated observation: "capability-based systems are the wave of the future and always will be." This proved to be a good example of the adage. The conceptual system had definite theoretical problems, but by and large, it looked as if it could be shaped if that company could afford enough work to meet the TCSEC B3 requirements. It looked as if A1 was going to be out of the question, no matter what. And so we worked on that until the company itself reorganized and the project lost its funding. About the same time, Lockheed, in Austin, Texas was competing for a government contract and I remember after that first meeting in Atlanta, I was routed not home, but off to Austin with Steve to talk to the Lockheed executives about our consulting to them on how to structure their design for a specific proposal that they were working on. That was an exciting proposal; it was a very sophisticated application, and I think we came up with a good approach. Those were the first two projects I got involved in and there was activity I was involved in three days after I joined the company. Steve had a remarkable set of connections and a large number of interesting — sometimes short term, sometimes longer term — consulting activities came up. I think on average I was involved in about four different projects of that sort, throughout the time that I was there. I was also involved in other kinds of consulting or research. But it was definitely for the computer security world, state of the art or beyond state of the art research all the way through. I had gotten involved in international committees when I was at NSA, chairing one of them, which I passed on to Carl Landwehr afterward. I continued interactions with various international groups throughout my tenure at TIS. We also got information security research contracts from foreign companies or governments. We got involved with a company called IABG [Industrieanlagen Betriebsgesellschaft MBG] in Munich, on their work at evolving what they called the Green Book, which was to be their evaluation criteria. The Green Book later became the exemplar for the evolving European's Common Criteria effort. This work had started while I was still at NSA but, because of strong relationships that had formed years before, we continued that work with IABG until they finished their research. Oddly enough, I got reinvolved with the IABG team, after joining ARCA because one of the studies that dated back to the days on covert channel analysis I'd worked on with Clark Weissman, Dick Linde in the mid- 1970s. In the beginning, no one believed covert channels were going to be significant. It turned out that during my time at NSA, the significance of covert channels became a little bit more pronounced when we discovered some faster channels than were mentioned in the Orange Book. By the time I was at TIS, potential speed of covert channels had gotten up into the tens of thousands of bits per second. And while at TIS, I had gotten involved in the security design of a system called PRISM, which IBM was selling and concurrently, one called MDF that Amdahl was developing, and one that yet another company — I think it was Hitachi — was trying to develop; these were all based on the same idea of hardware-based virtual machine monitor. The work with one of those three — so that I don't have to break the confidence —continued until I was at ARCA, in consulting trips that Chuck Pfleeger, and Terry Benzel, and I made together to that vendor. On one of these trips, we taught a class in how security principles worked on the one hand, but also we mentioned covert channels. During lunch, two of the hardware engineers who were designing the microcode, got together and raised an interesting question when we got back after lunch. We did a quick blackboard estimate and came up with a covert channel that was reliable and in the range of 10 million bits per second on that computer . . .

Mrs. Schaefer: Not bad.

Schaefer: . . . completely violating its isolation policies, executable by anyone. And 10 million bits a second is impressive.

Mrs. Schaefer: Certainly is. [Laughs.]

Yost: I remember in interviewing Steve Lipner on the A1 effort they had going at DEC…

Schaefer: Which we were consulting on.

Yost: . . . in the 1980s and covert channels were one of the big challenges.

Schaefer: Some of these issues had lives of their own. Getting back to the international projects, IABG was interested in a contract they had with the German intelligence agencies. And one of the last things I did at ARCA was a study that Chuck Pfleeger, Gary Grossman and I completed and delivered in Germany on convert channel control and estimation. So a lot of the research just continued, I guess, and because of work I had done previously, I was always involved in it. One of the others that accidentally started when I was at CTA — really my only research at CTA that ever resulted in a significant long-term study — was a multi object-oriented database management system that a company called ONTOS was interested in producing. I had begun consulting on this during my last year at CTA, and continued through all of my service with ARCA. We published a paper that appeared in the Oakland Security and Privacy Symposium proceedings on our object-oriented data management security model. We also ran into trouble with our old friend the TCSEC all over again when one of the contract monitors at Rome Air Development Center [which by then might've been called Rome Laboratories] discovered a conceptual problem about individual accountability and associated problems with audit trails in the context of accessing a database. The question there was at what level of granularity do you actually have a security-related event? Formulation of the query? The actual search? Or retrieval of results? Do you have a security event in the event that the results returned are empty? The answer is Yes, because if you ask the negative of the question and get a response that's negative, you've just learned the positive. In any query, is the entire database and its metadata accessed, or only the part relevant to the retrieved result? This ended up being a very rich problem, and again, it was the notion of what an object-oriented database is, the fact that object-oriented data management is almost exclusively implemented with programs that are written on the fly by users, and which, therefore, may or may not be security relevant depending on their own content. And all of the asynchrony and synchronization problems, and other interactions that you would think of may be relevant. So, this is hard stuff, interesting stuff. We came up with I think a good security model but unfortunately the ONTOS went out of business for unrelated reasons before the prototype could be implemented as a product.

Yost: This has been tremendously helpful. Before we conclude, are there any topics I haven't brought up or things you'd like to expand upon before we conclude?

Schaefer: Just one, and I think it's really a very important theme; and that's that I think mathematical training is extremely important attribute my success to this and I think extremely important to people who try to work in this field. NRL's John McLean is correct, and from the time that he introduced the preposterous System Z on, questioning the validity and adequacy of security models, as well as what they have to omit, is a very important part of all of the discussion. But I think that the other thing that is often missed is the importance of having a discipline to work with in identifying strengths and weaknesses of systems. You had quoted someone in the note you sent me, about whether there is any value in penetration studies. It's unclear to me whether there's much value in it today, but there certainly was when we were initially conducting the penetration studies because that's when we started learning what goes wrong, and deriving principles that could be used in general to come up with design strategies that are more robust. A significant part of my maturation was a benefit of having taken a course in programming that was taught by Edsger Dijkstra. A couple of us at SDC — Tom Henke and Sue Rho in particular — were privileged to take this week-long class, where we learned the discipline of programming with what are called weakest preconditions for correctness. It was before I took that course that I realized that I was applying this concept when I was doing penetration studies. The idea, basically, was to come up with what was meant by the notion of security and trying to determine preconditions for achieving that state, as a formal relationship among state variables. When you come up with preconditions for that, the next question is what is needed to get those preconditions established? And so one pushes backwards in logical time to find out what those preconditions would be. At some point, you can expect to discover a precondition that you can't come up with a logical derivation for. That is almost always a security flaw. That was the way that I identified essentially everything that I didn't accidentally trip over when I was involved in a security study. And it was also the principle I used when we were trying to derive architectures that might work. Again, coming up with a formulation that we could write in formal terms, and then deriving weaker and weaker preconditions to see if we could establish that state. That's a mathematical, well, logical process. I think that it's one that is hard enough that most people give lip service to it or have never been exposed to it in a formal class, but I really think that if we're going to achieve any form of security it's the principle that's most important to working in the field. I don't know much about what's been going on in current security environments largely because they seem to be completely unconstrained. Yet I feel that if one is going to be able to establish any form of reasonably secure network security, or PC security, there's going to have to be a mathematical breakthrough that's much, much stronger than any encryption or encryption technology. That literally constrains what can be done on a system but still allows people do what they need to do on the system. The catch is that every PC has its own system administrator; it's the person who decides what to put on it. And almost universally, those people have had no training whatsoever in security. As a result, whoever is going to try to design something that will actually work in securing networks is going to have to address the problem in a very different way than has ever been done before, and without a discipline, that's just not going to happen because I think the problem is basically unbounded. I can put anything I want on my computer and if I give that to someone else and they don't know how to decide whether or not it interferes with what works on their machine, they're likely to have a serious security vulnerability. This doesn't require my doing what I'm doing, maliciously; it requires instead that what I give them works with what they have in a way that's compatible and non-contradictory. Computer systems are logical systems and if you end up with a contradiction in the system, the system can't deal with it and, of course, in any logical system, if you have a contradictory state, everything can be derived, including any statement and its negation. I think it's a completely unconstrained environment that we're now working in that is going to prove to be the hardest problem to overcome. But I think that really is the synopsis of everything that I've been able to achieve or not achieve has been based on my mathematical training, which as it turned out, never had a single applied mathematics course; it was all theory. And I think somehow, people with that particular designated training are needed in this industry. I think that's pretty much it.

Yost: Well thank you so much for your time and your insights.