An Interview with

WILLIAM H. MURRAY

OH 433

Conducted by Jeffrey R. Yost

on

24 September 2013

Computer Security History Project

New Canaan, Connecticut

William H. Murray Interview

24 September 2013

Oral History 433

Abstract

In this interview computer security pioneer William Murray begins by discussing early work experiences and influences (his father was an IBM CE and manager, and his mother was a keypunch operator). The bulk of the interview focuses on his work at IBM in computer security and his reflections on developments in this field. This includes efforts with computer security at IBM SHARE, Bob Courtney as an early leader at IBM in this field, Horst Feistel and the cryptographic research group at IBM, MVS TSO, IBM's MVS Integrity Commitment, TCSEC, and RACF. He also provides context to a number of his publications including his influential Access Control Facility for AAS and Data Security and Controls. Murray was an influential figure with ISC-squared and the CISSP security credential and the auditing and forensics sides to security (working as a consultant for Deloitte & Touche and Ernst & Young after leaving IBM).

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 Charles Babbage Institute of the University of Minnesota. I'm here this morning on September 24, 2013 with William Murray and this interview is for CBI's NSF sponsored project "Building an Infrastructure for Computer Security History." We're in New Canaan, Connecticut. Bill, can you begin by telling me where and when you were born?

Murray: I was born in Washington, D.C. in 1935 at the Columbia Hospital.

Yost: Did you grow up in the D.C. area as well?

Murray: No, my father was with IBM and so I spent my preschool years in Atlanta, Georgia. I went to grammar school in Jacksonville, Florida; high school in New Orleans.

Yost: I understand your mother was a keypunch operator?

Murray: My mother was a keypunch operator. My father carried a tool bag for IBM. My uncle was a branch manager. So I was sort of predestined for all of this. However, of my 11 siblings, there are probably more lawyers than IT folks, but there are a lot of IT folks.

Yost: And your father was a customer engineer, an IBM CE?

Murray: Started out as a CE, yes. [He] went on to become a salesman and branch manager.

Yost: And he went on to be promoted to manage the state of Florida for IBM?

Murray: Yes. During the Second World War, my father was not in the service because he already had six children and so he was not eligible for the draft. Most of the IBMers at that time were accepting commissions. My uncle accepted a commission in the Navy, but my father managed the state of Florida during the war because he didn't have to go in the service. But his contribution to the war effort was that he handled most of the shipyards that were building the Liberty ships in Jacksonville; Savannah; Pascagoula, Mississippi; and all along the Gulf Coast to New Orleans. He was probably the premier application person on how you build Liberty ships. The Bill of Materials was essential, and managing that Bill of Materials was the start of the modern database. The next big thing like that that came along ended up in being the IMS database. That was put together to manage Bills of Materials for the space program. Rockwell.

Yost: So your parents obviously had an influence on your developing interests. Were there others?

Murray: Yes, my first job in IT was when I was seven. One of my father's customers was the Swisher Brothers cigar factory in Jacksonville. And my job was to pull the second card in a group with an "x" punch in column 80. I lived long enough to wire a collator to do that job, but on that particular Sunday, that was what I was assigned to do. My next IT job was when I was working at the Jackson Brewing Company, when I was in high school. My first job in the Brewery was in the mail room as a messenger. I still trade on the things I learned in that job. I learned about banking, accounts payable and receivable, payroll traffic, inventory, and distribution. I learned about business documents, transactions, and controls.

They moved me from the mailroom to the room where they were doing all the sales analysis; one took the sales slips off of the trucks and posted them to index cards by retail outlet. The IBM salesman sold them an IBM system to do that job. Since I had been a babysitter for his kids, he knew me and brought me in as a sorter operator.

I was absolutely fascinated by the fact that the job-to-job controls always balanced. I was talking about it at dinner one night [and] my father said, son, those machines are absolutely amazing. They're very reliable and very accurate. But check on them. Little did I know that he was determining my career, because I've spent an entire career checking on computers and trying to make sure that the results that they produced were the results we intended.

After flunking out of college — which is great credentials for going into IT these days — I was hired first by American Cyanamid as a lab technician in a hydro-cyanic acid plant IBM was hiring was hiring and training customer engineers to work on SAGE. The hiring manager in New Orleans was a fishing buddy and I applied. Because he knew that

I intended to go back for my degree and these were permanent positions, he would not hire me. Instead he sent me to interview with IBM Research in Poughkeepsie. Based upon those interviews, aptitude tests, and his glowing recommendation as to my character, I got the opportunity of a lifetime.

I went to work at IBM in the Poughkeepsie research facility on June 8, 1956. Thomas J. Watson, Sr. was to die on the 19th, but it was not my fault. I worked with Nat Rochester and the people who had developed the IBM 701. In fact, we had a 701 installed there. While I was there we went to a 704; we had a 705; we had a 650; and I got to work on all of those machines.

For training and assignment, I joined a cohort of recent college graduates. In those days, a great deal of computing was punched card oriented; that was where the data was. We also used punched cards to record and input our programs. My experience as an operator gave me an advantage and made me more valuable to my associates than I might otherwise have been.

Yost: Where did you go to school?

Murray: I went to LSU. And so when I [pause]

Yost: And had you established a major?

Murray: Well, it turned out that I was a very unsuccessful engineering student. So in 1958, I was going to be drafted; in fact, I was drafted. I went into the service for two years. My assignments in the service included a machine records unit and machine accounting in a post headquarters and a corps headquarters. When I came out of the service I went back to LSU. I went back into engineering but I stayed there for just one semester. One of my fraternity brothers talked me into going to the business school and I went from being on scholastic probation to the Dean's List in one semester. That pretty well demonstrated that I had made a better choice by going into business. One of my favorite courses was Accounting 2, which was controls. Back to that theme again. I only took Accounting 1 and Accounting 2, and Accounting 191 and 192; 191 was punch card accounting, and 192 was computer accounting. So those were the four accounting courses that I took and while I was there.

I had a student job in the computer center, and I had another job with what was then Standard Oil of New Jersey, the Esso Refinery in Baton Rouge, where I worked unsupervised on an early morning shift. Yield and cost data came in from the refinery in marked sense cards. My job was to convert the mark sense cards into punched cards and do the initial processing on those. I had a fabulous time; had great mentors; learned a great deal. One of my mentors on that job was Charlie Middleton. Charlie taught me to always over control a new application. He said, if you've over controlled, you can always relax the controls after the application is up and running, but if you under control, getting it back into control is almost impossible. So always over control a new application. That rule served me, my clients, customers, and colleagues very well throughout my career. I've had great mentors throughout my career and been very fortunate in that regard.

The last summer before I graduated, I worked for IBM teaching 650, 1401, and 705 programming. When I finally graduated from LSU I went to work in the IBM branch office in Baton Rouge, where I called on state and local government, finance, and health care.

Yost: And what year was that?

Murray: I graduated from LSU in January 1962, finally.

Yost: When you worked at the computer center what types of things did you work on?

Murray: I worked on programming and I was in support of the agricultural economics department, where they were doing statistical analysis. I was programming the 650 to compute standard deviations on what were then fairly large amounts of data. It's hard to imagine today, when you can carry a terabyte in your shirt pocket. A gigabyte in punched cards would fill a boxcar, and you couldn't read it in a year and a half. I had one client, would've been in the 1990s, who had a six terabyte database and the big problem was how to back it up because in the 1990s that was so much data that there was not time enough over night to back it up. So we had to learn how to break that down so that they could have a reasonable kind of backup in case of any kind of a disaster. So, I stayed in

Baton Rouge for five years. And then I went to White Plains to work on IBM's Advanced Administrative System, where I rejoined the cohort that I had worked with in Poughkeepsie.

Yost: And that was the Advanced Product Planning Group?

Murray: Yes, that was the Advanced Product Planning Group. They went on to do the American Airlines Sabre System and when IBM decided to build its equivalent of the Sabre System to handle all of the orders for the 360, that group of people got that job and I went back to work with them. The first project that I managed we called the "Bit Byte Fixer." It was a low level program that application managers could use to make fixes to the database.

The biggest thing that I had while I was working there was the security subsystem for the advanced administrative system. Today called access control, but in those days, it was all the security that we knew.

Yost: And other than that, had you had any exposure to pioneering security work to that point?

Murray: Well, from an enterprise perspective, we were pioneers. We pioneered end-user identification and authentication, list-based access control and administration. We invented it; no models. Our work was a model for many later systems.

There had been a lot of stuff going on in the laboratories. Probably the guy who was leading all of that, and who was in those days called the IBM Security Manager, was Bob Courtney. Unfortunately, Bob is not available to be interviewed these days. He's suffering from severe arthritis and Alzheimer's. But he was the manager of data security for the development division; what was then called the Systems Development Division, where most of the product development was being done. He got that job because he was the original product manager for the IBM 2260, which was an early terminal. NSA was concerned about the fact that it emanated EMT.

They were always concerned about leaking of data. They also worried about the leaking of data from mainframes. But the mainframes were so noisy that anybody who could recover data from them must have been really, really good. But anyway, Bob was the guy who was talking to them, so he was doing a lot of work in security. He was the guy who made the determination that you could not do automated teller machines without encryption, and IBM then hired Horst Feistel who did Lucifer for the ATMs. And that was IBM's earliest entry into cryptography.

Yost: He was part of the research group?

Murray: Horst was in the research group. He was in research in Yorktown, and Bob was in the Systems Development Division in Poughkeepsie. Horst Feistel then went on to work on the Data Encryption Standard (DES) with Walt Tuchman, et. al..

There's a lot of controversy around the development of the Data Encryption Standard. Harry Daniels was the guy at NSA who made the decision that let that effort go forward under the National Bureau of Standards. Ruth Davis was the director of the National Bureau of Standards. She recognized at that time, in part from the ATM endeavors, that Commerce was going to require some kind of encryption. NBS put out a request for proposals and IBM's proposal, after some negotiation, was the DES.

Part of the negotiation had dealt with hardware implementation. NSA did not want software encryption because they understood that that would turn every computer into an encryption machine. The negotiation revolved around being sure that the DES would be implemented and shipped only in hardware. IBM understood the agreement a little bit differently. IBM understood that it had to be in hardware but it could also be in software.

The hardware implementation needed to be on a single chip and that's where the 64-bit block came from; the fact that it could fit on a single chip. Negotiation over the key length was a little different. The NSA was willing to consent to a 56-bit key so the key was 56 key bits and eight parity bits. There was some question about whether or not the parity bits were of any use to NSA but my suspicion is they probably were.

Perhaps one of the more important things to come out of that whole effort was the idea of automatic key management. The IBM papers that came out at the same time as the DES was announced that described automated key management were incredibly important.

One illustration is the German Enigma effort. The Germans encrypted all of the traffic for an entire day and an entire theater of battle under a single key. If you recovered that key you got an awful lot of information. Of course, even recording all the traffic was incredibly expensive and when you recorded it you didn't know if you were getting the key or not. But the fact that they had encrypted so much information under a single key made the effort worthwhile and, in fact, it turned out, ultimately efficient. In modern cryptography, we don't do stupid things like that, instead we use a key for a single object, message, or session, and then we throw it away and use another. Therefore, little information is ever recorded under a single key and that makes cryptography dramatically more efficient than it had ever been before.

Yost: And what was your role with the effort?

Murray: I was the popularizer. I was in the marketing division and so my job was to write the support information that explained the whole idea. Remember, up until the early 1970s the very idea of cryptography was classified. There was no public dialog about cryptography; there was nothing in the academic papers about cryptography; and so the whole idea of it was novel when it came out. And to make people understand what it could do; what it was going to be useful for; how it was going to be implemented; just to explain the products and be able to produce the marketing materials that went with those products was my job.

By that time I had been moved from the Advanced Administrative System to market support. During the development of the access control facilities for AAS, I had learned to talk about security. There weren't very many people in the business at that time who knew how to do that. I had made lots of presentations and written on it. When it came time to get somebody to support the marketing effort I was a natural choice. I moved from in-house administration to marketing. My job, on crypto was to write the marketing materials; to understand cryptography, its strength, and its uses, and then to convert that understanding into language that customers were likely to be able to understand. They were very unlikely to be able to understand the cryptographers talking about their own work.

My two masterworks, as I look back on it, were the Access Control Facility for AAS and a manual called *Data Security Controls and Procedures*, which was published in 1976. Price Waterhouse had written a manual for IBM that told managers how to control their punched card applications so as to both ensure and demonstrate that the results were as intended. It was both technology specific and very dated. We need a similar document to address magnetic tape, disks, and terminals.

Like the crypto work, *Controls and Procedures* was to put technology in useful language, to talk about security and the application of it to the computers of the day, so that people could understand how they could make sure that the information that was produced by their computers was what they intended for it to be.

It was a harder problem in the day than people understood. [Laughs.] That publication stayed in use for about 20 years. I think there were at least three versions of it. I am particularly proud of that. One of my objectives was for it to be sufficiently technology free as to have a long life.

Yost: And it wasn't tied into any particular system.

Murray: I didn't tie it to any system but dealt with information in the abstract to give it a longer shelf life than the Price Waterhouse document had had. And as I say, it stayed in publication for 20 years so I was reasonably successful in achieving that. Of course, I produced a lot of technology specific materials. For example, I produced a little manual called "Good Security Data Security Practices." Part I addressed programs, libraries, programming, programmers, execution or use, and users. However, Part II addressed specific example systems or user interfaces like the MVS Time Sharing Option (TSO) and the Remote Job Entry system (RJE). Each of these got a dozen or two specific recommendations, no more than a paragraph in length and based on six propositions: 1) risk is lowered when multiple people are involved in sensitive functions; 2) risk is minimized when each individual has access to only those resources required to do their jobs (rule of "least possible privilege"); 3) sensitive activity should be independently authorized; 4) duties should be assigned in such a way and records kept such that individuals can be held accountable for their actions; 5) duties should be assigned such that one person doing his job acts as a check upon those in his environment and is checked upon by them; 6) no one should have access to sensitive combinations of resources. I think I may write a book.

Yost: When that came out in 1976, what were the initial responses from the audiences it reached?

Murray: My recollection is that it was very good. As late as the early 90s it was still considered a material part of the body of knowledge. In1976, IBM's bestselling publication was still 1401 Principles of Operations, which remained on the bestseller list for much longer than the machine lasted. Of course, there are still a couple of operating 1401s in the world. Two of them are in the Computer History Museum. One of them came from just south of here in Darien, where it was used to do country club billing until somewhere around 2000, and then the machine was given to the Computer History Museum. So there are two of them out there that are still operating. It was a very influential machine in its day and I got a lot of early experience talking about things [by] teaching programming. I taught programming at LSU. I was the only undergraduate who had full professors in his class. That was computer programming, that was 650 programming. But I also taught; the summer before I graduated in January of 1962, I worked for IBM in the Mobile, Alabama, branch office teaching programming. I taught a 1401 course and two 650 courses in Mobile, and then for August, I went to Dayton, Ohio, to the Wright Patterson Air Force Base, where I taught 705 programming for the Air Force Logistics Command.

Yost: Wright-Patterson was headquarters for Logistics for the Air Force, correct?

Murray: Right. And it was also a SAC base. They were still flying B-52s. Yes, they could've still been B-52s.

Yost: What was the time span of the development of the access control facility for AAS?

Murray: 1969, 1970, 1971. The philosophy of AAS was unique for its day. While it was built from standard IBM products, it was purpose built for a specific set of applications. It was ultimately going to have 5,000 users in branch offices all over the U.S., and they were going to be processing all the business applications: accounts receivable, the billing, order entry, all of those applications. Even education enrollments were going to run on it. In fact, that was one of the applications that I did.

Yost: And these were customer facilities, this wasn't IBM offering service?

Murray: Right, this was IBM's management of its business, not offered to customers. IBM's management of its business and the 5,000 customers were all IBM employees. Ultimately, that user population grew to about 10,000. So we had to come up with a user identification and authorization scheme. We had to come up with an access control system. It turned out to have been list-based. A user was given access to transaction types and he could have a very long list of possible transaction types. Transactions tended to become more granular over time and therefore, the number of transaction types went up probably higher than we had originally expected. One of the interesting things we had was that there was a tree structured set of menus that led the user from logon, through the name of the application down to the transaction type that he wanted to process. The transaction also had a unique four-character name; we expected them to rely on the tree structure in order to translate from what they had been doing before to the transaction type on the system. After a couple of years we found out that they no longer thought about the business in terms of the way they had been doing it before. Now they thought of the business in terms of the four-character transaction name. They no longer used the tree structure at all.

The access control facility ran at the application layer, and it was safe, in part, because development was not done on the operational system. There was no possibility for; there were no compilers, there were no assemblers, there were no linkage editors. None of those things actually ran on the production system. They all ran on a separate development system. So there was no programming done on that system. One of the briefings that I did was for Willis Ware. Willis came in for a briefing one day and I presented to him this system; and I was very, very proud of it. And when it was over, Willis said well, it's very interesting but it doesn't address the general problem. I said Dr. Ware, what is the general problem? And he said the World-Wide Military Command and Control, which was the biggest security RFP that had been put out at that point. And I said, it looks to me like it would be well adapted to a military command and control system. He said well, you've got this sergeant someplace who's writing a program and executing it. I thought at the time, you know, that's a really strange way to do a command

and control system; the last thing I want happening on a command and control system is somebody writing and executing arbitrary programs. I would want it to run like an application system, like the advanced administrative system. It was years later, when I was finally recognized by Willis as a peer and a colleague that we had the discussion about what generality and flexibility one is willing to surrender in the name of security. You have to give up something; you cannot reserve all of the generality and all of the flexibility that comes out of the factory to every user and still be able to say anything meaningful about security.

I've always been an advocate of application layer security, because at the application layer the intent of the system is much more obvious. Steve Jobs agrees with me. In Apple's iOS the system looks to the user like a collection of purpose-built applications. At any instant, he is using a single purpose-built application. In the IOS system he is reasonably confident that that application is not being interfered with by any of the other code that's stored on the same system. However, at every other juncture, in the history of computing, where we have had the choice between generality, and flexibility, and features, and functions every time we've had one of those choices — and security on the other hand — we've opted in the direction of generality and flexibility. The government, which protested that it wanted secure systems, trusted systems, never operated them that way. In the late 1970s, NSA came to IBM and said they wanted a B1 MVS system. We argued that if they really wanted a B1 system they ought to let us build it on the AS/400 architecture, where object properties were reliable; where if you gave something a label you could rely that the label wasn't going to be tampered with. Even on modern systems, while the properties of an object may be hidden and not normally displayed, they are still modifiable, malleable, on that system. If I open a Microsoft dot-doc file, I will not see the properties but they are there and I can manipulate them if I want to. If I had assigned a label as a property, a classification like top secret, it wouldn't be reliable.

It turns out that in order to make things completely reliable on any system, it's got to be a closed system. What the government didn't like about the AS/400 is that it's closed. It is also still one of the best security architectures that's in the marketplace. It's the only one that's really survived. All the rest of them have failed in the marketplace because they hid too much of the generality and flexibility from the end users. In fact, that was the reason that IBM's FS architecture was not the successor to the 360 architecture. Instead of replacing the 360 architecture with the FS architecture, which went on to be the AS/400, IBM simply improved and built upon the 360 architecture. And the 360 architecture survives until this day. They both survive, side by side, but IBM was never willing to try and sell a closed system into the open system marketplace. So when NSA came to us in the 1970s and said we want a B1 MVS system, we said we think you really want a B1 system, and if you really want a B1 system you ought to let us do it on a; and they said no, no, it has to run the MVS job stream. At the time, the MVS job stream in punched cards stretched around the world three or four times, out to the moon and back about nine times. It included all kind of disorderly code. It was extraordinarily difficult to do a B1 system that didn't break any of the applications that were in that job stream. I argued for not wasting time and money and I thought that I had carried the day.

NSA went back to the Department of Defense and they went to all their aerospace contractors and said we're going to require that you run on a B1 system. So all the aerospace contractors who had jillions of dollars invested in IBM application programs came back to IBM and said we gotta have a B1 system. And finally the pressure was so great that IBM said okay, we will skip an entire annual release of MVS, dedicate the time and money to creating a B1 MVS system. Of course, in order to be reliable as a B1 system, it had to operate as a closed system. Whereas for off-the -shelf MVS management had to control five things, for MVS B1 they had to control nine. This included the content of the system so that object names would be reliable, like they are in in Apple's iOS.

The cost was extraordinarily high, both for the development, and for skipping the release — there was no new device support done that year — nothing was done except making it a B1 system. [It was] announced to great fanfare. Everybody was thrilled. We went to the Computer Security Conference, and to SHARE, and GUIDE, and presented it; and everybody thought it was absolutely wonderful. The only problem with it was it didn't sell. I was thrilled when I heard that the sales had gone up by 50 percent, until I found out that that was from two to three. I think there were ultimately five sold, and none of the five was ever operated as a B1, i.e., closed, system.

Yost: And when did it come out?

Murray: Let's see. RACF release 1.9. It was 1.9, I remember that much. I better do it on that system there. There we go. I think it was probably around 1978 but Wikipedia may know. Actually, more important than RACF or the B1 system was probably the MVS integrity commitment, which said that except by means of a mechanism under management's control, a user could not expand his privileges and the [pause]

Yost: I have something that Eldon Worley gave me that shows Version 1 Release 9 in September 1990.

Murray: Was it really that late?

Yost: AS/400 was in the mid-1980s when it came out, and The Orange Book criteria wasn't published until the mid-1980s—first in 1983 then refined and republished a couple years later.

Murray: Okay, time flies when you're having fun. I serve on the board of the IT History Society and one of our themes is had we known we were making history we would've taken more pictures.

The MVS integrity commitment was the first real security warranty and it was very powerful. Essentially what it said was if you could discover a way in which a user could escalate his privileges, by a mechanism not under management's control, IBM would fix it. Now that's a limited remedy; people would've liked to have better remedies than that, but IBM still sticks to that commitment after all these years and it really did contribute.

In the early days of MVS there were tens of security problems, what we then called integrity problems. About half of them were identified inside IBM. The other half were identified by about five or six customers, including Sun Life of Canada and the AWRE in the UK, who devoted a significant amount of resources and effort to finding them. The system programmer at Sun Life who did that work ultimately left Sun and went to work for IBM as a consultant discovering integrity flaws in MVS and helping them fix them. IBM's attitude toward it was we never tell people what the problem was, we just tell them we fixed it. We don't want to leak any more information about the nature of the problem than is absolutely necessary. But they came from a relatively short list of types of problems, most of them identified in the RISOS project that was done by Jim Anderson and Bob Abbott. They identified a relatively short list of problems that are still the source of most of our problems today. It included what they called "race conditions" and IBM called "time of check to time of use (TOCTU)." Topping the list was, and still is, unchecked inputs or incompletely checked inputs. For example, an application input might contain an "escape character" or command sequence that caused the application to pass everything that followed it to the environment. This is the class of vulnerability exploited by buffer overflow and SQL Injection attacks.

It turns out that that problem has gotten harder over the years, in part because we layer software so heavily now. The program that accepts the input may not even know what layers are below or what inputs are acceptable to them. It may accept and the layers below may have escape mechanisms in them. That's what a SQL attack is, okay? I'm able to put a SQL command into an input field. The application thinks it's an address field, right? But unless it screens for commands an attacker can put a SQL command into that field and can see more of the database than was ever intended. The problem is aggravated by the fact that so many applications today rely upon the application to protect the database, rather than the other way around. We ought to be doing access control and input checking at every possible layer but we don't. That's a big part of the cause of our problem. An attacker can put in something that, to the application, looks perfectly legitimate but when it gets processed by something down below, it has the potential for database leakage or program contamination. An application that does not constrain the length of a field, may overflow the space allowed for it by a process below in the environment and overwrite code. Look at the problem that Willis Ware wanted to solve, that is to reserve all the generality to all of the users and still say something about security. He worked for von Neumann. One of von Neumann's claims to fame was the idea of the program as data; that is, a program could operate on itself or some other program could operate on it. What that does is allow for a program to be contaminated by its data; and if the input is not properly checked, we run the risk of that and so we get things like buffer overflows.

Yost: The von Neumann architecture itself establishes a point of vulnerability?

Murray: Yes, the potential for a process to be contaminated or corrupted by its data. Until von Neumann computing machinery used what we now call "typed" storage, one type for programming, for example, a plug board, and another, for example, punched cards, for data.

Von Neumann didn't want to make an early choice about how storage, what he called "a precious resource," was going to be used. Von Neumann didn't want to have to make that choice; he wanted the storage to be used as easily for data as it was for procedure because it was terribly, terribly, terribly dear. It was incredibly expensive. That condition only lasted for a decade or so; well, a generation; it lasted about 20 years. By the mid-1970s, the cost of storage was starting on the trajectory that it's been on ever since; it was dropping dramatically. And so in 360 architecture, although the reservation was still there, we no longer modified programs; we wanted them to be re-entrant. Let me give you another illustration. On a 1401, a punched card machine, originally; all the card machines were going to read punched cards. The op code for "read a card" was a "1." Every program was going to have a "1" in it somewhere because it was going to have to read a card. So whenever you read in a constant "1" what you did was you referenced the location where you put the punched card, the read a card instruction. The op codes for "branch unconditional" and "no op" — do nothing — were deliberately chosen in such a way that by adding the same constant to "1" you produced the other. And that was how you did switches. So switches were always in line in the code. Again, the idea was that the storage was dear. Nobody remembers this, but on the day of announcement for the 1401, you could buy it in two models, two different storage sizes. One of those was 2k.

When I ask an audience of 600 or 700 IT people, what the other option was, they all say 4K. And I say no, the other option is 1.4K — 1.4K and 2K — and the machine took its name from the 1.4K of storage. That's what 1400 meant. Fortunately for IBM, they didn't ship many systems like that. By two and three years into the product life, everything shipped was 16K. But literally, on the day of announcement, the two options were 1.4 and 2; so storage was very, very dear and we did very, very strange things. By the time of the 360 we had 16 megabytes of address space, but the early machines, my recollection is, they shipped with 64, 128, 256, 512K of storage. And this was before virtual storage. Virtual storage had been invented at that point, but the only machine to implement it was the B5000. As a consequence, the B5000 was slow; didn't scale very well at all; but it was brilliant architecture. As the circuits became faster and faster, and backing store cheaper and cheaper, virtual storage became more and more practical. And so we very soon ran out of the address space in 360 architecture and had to extend it from 24 bits to 64 bits.

Note that virtual storage offered us some process-to-process isolation. An address referenced by a process was always interpreted relative to its own virtual address space. It had no way to reference physical storage or the virtual storage or data of another process.

So when storage was dear, the ability of the program to operate on itself and to use storage efficiently in that sense, was very important. But it didn't last very long. In a world in which a gig sells for a dollar, the von Neumann architecture persists but is hardly even useful, much less necessary. The preference for generality and flexibility is the source of about half of our security problem. Complexity accounts for the other half.

In an AS/400, one cannot execute data or change programs. Programs can be replaced with something with a similar but more qualified name, but you can't change them. And things whose properties are data [that] can't be executed. The AS/400 represents a choice of security over generality and flexibility. It has a limited market. Most of my students have never even heard of it, much less appreciate it.

Yost: Some of those; you mentioned that early meetings with Willis Ware, presumably that was in the early 1970s?

Murray: That was in the early 1970s.

Yost: And in 1970; the Defense Science Board Report that Ware put out, were you aware of that at the time and did that have any immediate influence you were aware of at IBM?

Murray: I was not aware of the work at the time.

Yost: And what about in the early to mid-1970s, the work that Roger Schell was leading?

Murray: I was aware of all that; I was aware of all the stuff that was being done at M.I.T. IBM was funding part of the work that was being done at M.I.T. Project Mac was being funded partially by IBM. We were aware of all of that. Of course, we were aware of the public e-cryptography work that was being done at M.I.T. at the time.

Yost: Did the security design of Multics have any influence on IBM?

Murray: It had quite a bit on IBM. I wish it had had more on the industry, as a whole. Multics' biggest legacy today is UNIX and a large portion of our problems deal with the fact that a machine that was intended to be a single user system has been scaled up into a multi, multi, multi user system. I think one of the reasons why the C level in the trusted computers; in the TCSEC did not require list-based access control, was to accommodate UNIX. It was very clear that most of the applications for C2 systems were going to require some kind of list-based access control.

It's interesting that the SHARE Security White Paper, which drove IBM's access control efforts, was done by systems programmers for systems programmers. And they wanted a rules-based system. When RACF came out and it was list-based, they weren't very happy with that. So that was the origin of ACF2. Barry Schrager, who did ACF2 with his colleagues, was the guy who led the SHARE Security White Paper effort. I liaised for IBM to that project at SHARE. In general, rules-based access control, as in ACF2 and Unix, offer better performance because they may not require any reference to data in secondary store.

Yost: I definitely want to ask you about that important 1974 SHARE meeting, but before that, can you tell me about the meeting in December of 1972 on MVS ACM Workshop on Control Accessibility in Rancho Santa Fe, California?

Murray: The hosts for that project, or for that conference, were Ruth Davis, Director of National Bureau of Standards/NIST, and Walter Carlson who was on leave from IBM to be the ACM President. A great portion of the planning for it was done by Bob Courtney, and as a consequence of that, I was invited. Had Courtney not been in the planning, I wouldn't even have known about the conference, much less been invited to participate. Not only did he get me invited to it, but he got me appointed to chair the auditability subgroup. There were three subgroups: one on integrity, one on access control, and one on auditability. I chaired the auditability section.

One of the things that I thought that was going to be most influential was the report of that working group. It said that every time control passed from one process or person to another that had to be logged on both sides. So if I was going to give you something I had to put in my log that I had given it to you, and you had to put in your log that you got it from me. And when you gave me something back, same thing had to happen. So if an application was going to make a request to the database manager, the application would have to log that. The database manager got the request, and it had to put it in its log. All this was to be designed and architected. Did not happen. We designed and architected access control in most of our systems. We never designed and architected auditability in those systems.   Courtney always argued that auditability was more important to trust than access control.

In the working group we defined auditability as a property of the system that enabled management to observe it and control it, that is, to exercise a directing or restraining influence over its behavior, use, or content. It includes the ability to determine, and fix accountability for how it got to its present state and what it looked like at any point in the past.

We have few systems out there today that meet that. We have some application systems where you can determine what the system looked like in the past, but we've got very few general purpose systems where you can do that. But anyway, that conference was the beginning of what I think of as modern computer security. Out of that conference came the IBM data security study, with the seven contract groups; out of that came the beginning of the Multics; well, I guess Multics was an input into the conference because Multics was already on the table. It was probably the first place the Mutics Trojan horse appeared in public documentation. Roger Schell was there, Steve Lipner was there, Willis Ware was there, Steve Walker was there, Blake Greenlee was there, quite a prestigious group of people, many from the industry side, many from the customer side. In fact, I think Blake Greenlee, at that point, was with a bank; not yet turned his collar around. He reformed twice. Once when he left the agency, and once when he entered the clergy. [Laughs.] Who else was there?

Yost: Do you know if the papers presented survived?

Murray: I think I've got a couple of late submission papers, but I don't have a copy of the original report. Every time I have wanted to refer to it in the past I've called Steve Lipner. Steve is a packrat and he's kept all that stuff. Periodically, I try to throw stuff out. I also collect more stuff than I ought to, but, you know, I can almost draw a line at the point at which I became a computer user because that's where my recordkeeping got a little bit better. But before that; and as I say, I didn't understand that we were making history at the time; only looking back on it do I realize we were making history. But Steve Lipner, I think has a copy of that Rancho Santa Fe report.

Yost: Did that meeting inspire putting access control on the program of the 1974 SHARE meeting?

Murray: No, I think that came out of; that was organic SHARE. Recognition of that requirement, I think, arose organically in SHARE from a number of different places. University of Illinois Chicago Circle Campus, where Barry Schrager worked was [pause]

Yost: So from customers?

Murray: Yes. You know Schrager would define security as protecting the system from students, but users in general, and it was becoming important as we were moving more and more computing and data on line. [pause]

Yost: That had been an issue with CTSS.

Murray: And that's why, you know, when the terminals came in, and online computing and online programming, particularly, came in, when we were doing AAS we were building an online system for users without programmers. Our programmers were still coding on paper and submitting it on punch cards. But we realized the direction that everything goes on line. At that point, Digital Equipment was doing the PDP-11s and it couldn't read punch cards. PDPs could read punch paper, mostly for programs, but they couldn't read punch cards. Ken Olsen asked why would anybody do that? Well the answer to the question originally was because that's where all the data was stored. If one looked at the early computer orders, they were orders to process data that already existed in tons and tons of punched cards. By 1980 or so, we had pretty much stamped out punched cards except for a few very rare applications. I remember going into a computer room in Dallas, and having to order a plastic cover to go over the card reader because it was so unused that it was all dirty and dusty.

I remember my sister telling me that (she managed a computer center for a steamship company) she managed to convert all of the applications from punched cards to terminals and disk drives; and her successor converted them all back again. Some people just really, really liked to be able hold the data in their hands. The idea that it was invisible was too uncomfortable. We used to do a demonstration in the early days; we would dip a piece of magnetic tape into a solution of carbon tetrachloride and little tiny iron filings.

When you pulled it out, you could see the bits as they appeared on the tape. It was like developing the tape. It really made some people a lot more comfortable to be able to see the bits and know that they were really there. [Laughs.] Punched card people really had a problem if they couldn't see the data.

The SHARE effort was organic within SHARE. And Guide had a similar effort going on at about the same time. It's hard to remember how big SHARE and Guide were in those days, but SHARE and Guide both got to the point where their two big meetings a year — they did four a year, two big ones and two little ones; the little ones were mostly committee meetings — were limited in the number of places that they could meet, a number of cities couldn't accommodate them. Now, of course, they got to be about 6,000 or 8,000 people, each one of those and there weren't very many people that could accommodate that many people. Today a city that can't take 60,000-100,000 is not a convention city. But in those days, that was a really, really big meeting. So it was Miami — couldn't go to Vegas because it was Vegas — Miami, New Orleans, Chicago, Boston that could accommodate a Guide or a SHARE meeting. But the requirements arose organically from customers. SHARE and Guide were their means of communicating their needs to IBM what the requirements were.

Yost: So there was a session in which Barry Schrager presented, Eldon Worley presented, and someone from Boeing. Do you recall that session and [pause]

Murray: I don't recall Barry and Eldon Worley being in the same room at the same time, but I'm sure it must have happen. I just don't remember it.

Yost: They both do, so [pause]

Murray: I am sure they probably were. I can probably remember many times I was in the room with Donn Parker. Donn Parker and I used to do; Donn always came from the computer crime side of things and I always came from the errors and omissions side of things. He always came from essential practices, and I always came from list-based management. We always met in the middle, but our rationales were very, very different and so we used to do a dog and pony show where he'd stand on one side of the stage and I'd stand on the other, and we would take questions from the audience. Each of us would address it from a different space, but we would end up meeting in the middle on what the likely output was going to be.

Yost: Eldon told me that Bill McPhee and Bob Courtney wanted that session to gauge customer responses to access control. Does that sound plausible?

Murray: That's highly plausible. Bill McPhee wrote the paper that says these are the sources of integrity problems in the operating system. Bill McPhee called it time of check to time of use, TOCTOU, where the system would check something to make sure it was in the proper place, in the proper bounds, but wouldn't rely on it for some period of time. And then when it relied on it, it wasn't the same as when it had checked it before. So you either had to check something immediately before using it, or you had to ensure that it was bound, such that between the time that you checked it and the time that you used it, it couldn't have changed. The concept of binding I've always thought was a very important concept; [it] is poorly documented and understood in the literature. I define binding as resolving and fixing so as to resist later change. You can do that in operating systems by putting things into protective storage. But the concept of [binding] and the idea that we bind things at different times. We bind some things at coding time, some things at compile time, some things at leakage edit time, some things at application time, but the concept is very, very poorly understood and documented in the literature anywhere and I think that's a source of some of our problems. But we solved a whole lot of these problems very early; we knew what the solution looked like. But again, every time we came to a junction we would opt for openness, for generality and flexibility.

Yost: And that was largely customer driven?

Murray: Yes. One of the ones was; we had the MVS operating system, which was the high end operating system. We had another operating system called DOS VSE. For MVS, we could make a strong statement about the integrity of the operating system; for VSE we made a much weaker statement. We said it resists accidental interference between processes, and accidental changes; it's not able to resist malicious processes or malicious changes. And Tom Bloomer, who was my boss in marketing support, kept pushing for integrity commitment for DOS VSE, a stronger statement that it would resist malicious interference. And so the VSE people went through an exercise to come up with what they might have to do to VSE in order to make such a statement. When they tried that on the user group, the user group said no way. Those things that you look on as integrity exposures, we treat those as features and there is absolutely no way we're going to give those up in order to get a stronger statement.

And we said well, you're going to have this potential for malicious interference. They said we solve that by running multiple copies of the operating system under VM360. We put application and its own operating system in a different virtual machine and they can't interfere with one another. So we can get process to process isolation without having to give up these things that we look on as features.

That's one example of customer-driven; another was Windows NT. If you go back and look at Helen Custer's book describing the security for Windows NT, it's gorgeous; I mean, it is just a beautiful architecture. Bill Gates was committed to it until his marketing people said, ps-s-st, Mr. Gates, if you do that, you're going to break all these legacy applications and you're going to break all the games. And so we saw the split between Windows NT and Windows ME. Remember windows ME? Everybody's forgotten it now, but that was the solution for that problem. For the enterprise customers, they built Windows NT but for consumers, they would ship Windows ME, which wouldn't break any of the games, and which would accommodate all the legacy applications.

A big part of the security problem has always been legacy applications. You want to do a B1 system? Well, we know how to build a B1 system. You want to build a B1 system that runs the MVS job stream, all those legacy applications out there that stretch in punch cards to the moon and back nine times? That's an impossible problem. We confronted those over and over; it left us where we are today.

Another contributor to the problem was the internet protocol. The internet protocol is completely and totally open, and completely and totally flexible, and completely and totally general. TCP/IP drove the IBM competing architecture, SNA/SDLC, which was very secure; TCP/IP drove it from the marketplace. And the reason was because SNA/SDLC was closed, and TCP/IP was open. If you wanted to introduce a new device into an SNA/SDLC system, you had to go through a provisioning step. In a lot of cases, the provisioning step couldn't even be done while the network was running, right? So ultimately, SNA/SDLC just couldn't succeed in the marketplace. It was secure, it was beautiful, it was well done, but TCP/IP drove it out of the market.

Bellovin and Cheswick articulated four network policy positions, paranoid, restrictive, permissive, and promiscuous. At the time paranoid, what we now call "air-gap," was the default policy for business. Restrictive, everything not explicitly allowed is implicitly forbidden, what we now call "white list," was the policy for SNA/SDLC. Next came permissive, everything not explicitly forbidden is implicitly allowed, think "black list," even that was not the policy for TCP/IP. No in fact it's "promiscuous," that is, not only is nothing forbidden but there are not even any controls available to forbid anything if one wanted to. Promiscuous is the Internet Protocol, it was and remains the default in the internet.

I think that one can make a case that the absence of security in the internet contributed to its adoption. One cannot make such an argument for the counter-factual, that is that it would have been equally successful with, for example, a restrictive policy.

What we do know is that, for security reasons, business did not want to connect to the early internet. They were driven to it by economic forces. The propensity for two networks to connect is a function of the number of nodes in each of the networks. As the number of nodes on a network goes up, its propensity to connect with networks nearby goes up as an exponential function of the number of nodes in the combined networks.

Think about when you could first make an international dial call, okay? The value of all the networks went up exponentially when you could begin to do that. Said another way, there are few things as useless as a disconnected telephone.

Business was led screaming and kicking to connect to the internet by that economic pressure. They gave up their paranoid policies reluctantly. Business tended to prefer a restrictive side on access control, and access control with restrictive rules. The internet was coming from the other place. It had been a community of friends, right? No malicious people in it, everything was allowed, there weren't any controls built in. So we ended up, even when we started to put in controls, the dominant policy in the internet is still permissive.

One interesting thing that happened was that at one point, there was great concern about what was going to happen to the internet when people started putting wireless networks into their homes. It was going to be terrible; now all the hackers are going to be able to use those access points to get into the internet. Didn't happen that way. In fact, what happened was the security of the internet got better when people started to put wireless access points in their homes because the wireless access points protected their systems from the internet, which is where all the hostility was.

If one connected a new, native raw system directly to the internet today, it'd be contaminated within seconds to minutes. But your neighbor is not trying to hack into your system on the wireless side. It's possible. It's a vulnerability, particularly if you don't enable the encryption, but it's not happening. So when we put in wireless access points, we effectively changed the access rule for those home systems from permissive to restrictive, which is where we should have put it in the first place. Cisco sells enterprise level network cards, which have encryption on the card, have key storage on the card, and have firewall on the card. Thus, when one connects one of those cards to the internet, one is not suddenly opening 60,000 ports. Instead applications open only those ports that they are using and controlling.

For a few years, I used wireless access points for hospitality gifts. If I stayed with you for a weekend, one of the things I would do is install a wireless LAN for you. I would stick the wireless LAN in between the existing computer and the hole in the wall. Now, when I did that, I changed the rule from permissive to restrictive. In all the times I did it, and I must have done it at least a dozen times, I never broke an application, which means that we really didn't need the permissive rule. Now we probably needed it at the origin. Would TCP/IP have driven SNA/SDLC from the marketplace if it had a restrictive rule? I don't think so. Openness was essential to its success. That was how the internet grew organically without any design or intent on anybody's part.

Yost: It grew early on in friendly academic circles and then gained momentum.

Murray: Right. But if you had had to provision everything that you added to the network, if there was some control someplace and somebody saying who could add a new node, or a new application, or a new this or a new that, or even a new protocol, it probably would not have succeeded as it did. It grew organically because it was open and permissive. Every time we have had a choice between open and permissive, and closed and restrictive, we've chosen open and permissive over secure. So my sense is that security will always be chasing the requirement; it will never ever be better than we absolutely need for it to be. The internet is probably secure for 80 percent of the applications that we do on it. For many of the remainder we have sufficient compensating controls but we will always have some applications running on the internet for which it is a hostile environment, too hostile for that application, but we will run it anyway. At the margins there will always be problems.

We are now at the point where many enterprise networks are so large that we can no longer say anything about the safety of those networks. NSA, Northrup Grumman, all the big aerospace companies, now operate with the assumption that there are comprised systems on their networks. We no longer control the edge, we no longer control the perimeter, we'd better focus on the center. We'd better focus on end-to-end security terminating on the application, not on the perimeter and not on the operating system.

Yost: The focus of the trusted computing effort of the federal government, of course, was on the operating system.

Murray: Yes. I suppose that it made a difference, but for a number of reasons the difference was not in proportion to the money the government spent on it.

We do not want secure operating systems, we want secure application systems; operating systems are merely a means to an end. We do not want not want "secure" operating systems, we want "securable" operating systems, operating systems that can be operated securely for a specific application and environment.

They thought they needed "multi-level," a system so rich they could use it simultaneously for "classified" and unclassified users and resources. However, historically and by definition, in no other environment did they ever mix classified and unclassified data. The motivation was they wanted to be able to share expensive hardware across users and uses.

Hardware did not stay expensive very long but they were so focused that when it got cheap, they failed to notice. Of course, we still have to share data in a controlled way but we use applications, database services and file servers to exercise that control. The problem was no longer one of sharing expensive hardware but of managing large populations of cheap hardware.

The effort began with an abstraction, called a kernel. The way that they conceived it, it would encapsulate both process-to-process isolation and the mediation of all sharing. Not only did this complicate the problem but caused them to reject a solution that we proposed where a hypervisor, VM/360, provided isolation, and a process, RACF, running in a virtual machine under the hypervisor, mediated the sharing. Even later, when they had expanded the abstraction to the "trusted computing base," they still did not like solutions that separated isolation and mediation.

One reason for placing all the trusted computing capability in one process was to facilitate early verification by inspection, independent of the application or environment. The purpose of this early evaluation was to remove risk from the purchase decision. This may have been both ineffective and over-constrained. Not only was the evaluation expensive but its duration approached the life of a version of the product.

Finally, the effort did not work because the market did not want a securable, evaluated, but relatively expensive, late, and closed operating system. The market wanted Windows, not OS/2, Linux, not AS/400.

Murray: I do remember the Eldon Worley/Barry Schrager thing because it was my job to disclaim Eldon's presentation. "This is not a product. It will never be a product." That was my job. My job was to introduce Eldon and to disclaim his presentation, that this is not about product, this is an effort, okay? This is something that's being done inside IBM, and it was. Eldon built it for his own mission.

At that point, the formal development effort was crashing and RACF turned out to be the back stop. It was second choice. We have a designed effort underway. That's right, I do remember that because my job was to introduce Eldon and disclaim that presentation; this is not a product presentation and this is not going to be a product. But when the other effort failed, it ended up being a product.

Yost: Eldon told me the original design was to protect all data, which Barry Schrager said his ACF2 did do, but that the original RACF did not, and that that was customer driven. Is that your recollection that it was a later version, and apparently there was a survey?

Murray: Yes. The default rule for ACF2 was restrictive. The default rule for RACF and the other access control facility, whose name I've now forgotten because it never saw the light of day, they were all intended to go into existing facilities and for the customer then to decide how to apply them. Now, you could configure RACF to be restrictive by default, but the default was it was open, and that was in fact customer drive. And that goes back to what I've said and have been saying all along, which is every one of these junctures.

The reason for the permissive default was to facilitate non-disruptive installation. When one installed RACF it had no effect until one defined resources and access rules.

Yost: So protection by default was the initial marketing point of ACF2 to differentiate it from RACF?

Murray: Yes but of course ACF2 also had a non-disruptive installation strategy. While it was restrictive by default, it had a mode setting ($MODE(QUIET|LOG|WARN|ABORT). Installed in Quiet mode, it was passive. In Log mode, it would make an entry for any access for which there was no rule but it would permit it. Warn was the same but it would also give a message to the user. Almost all instances of ACF2 were installed in Quiet or Log mode; legend has that many never got beyond Warn. When I was with Deloitte we proposed a $200,000 engagement to move a client from Warn to Abort. We didn't get the engagement but the client never got to Abort either

We ended up with three access control facilities. We ended up with RACF which was list-based; Top Secret, and ACF2. Each of them could do the whole job, but each of them had a different rationale. If you looked at RACF, the search argument for the database was the resource name and the function that was returned was a list of users and their permissions. If you looked at Top Secret, it too was list-based, but it was organized the other way. The search argument was the user name, and what was returned was the list of resources that the user could access. The default rule for ACF2 was that a user had exclusive access to every object that he created and that was qualified with his user name. In a college or university environment that was most of the data. The beauty of that rule was that its application was fast, it required no external reference.

Each of the three products had places where they fit better, were more applicable than the other two. RACF was best in an environment where things were being used across users, and being created and deleted frequently. The life of an object was uncertain; I would create it and it would be used by multiple people and then it would go away.

Top Secret was in its element when the list of resources was relatively constant, it didn't change, but the users were coming and going. So in a bank, I've got a limited number of resources, transaction types, very stable; but I've got new tellers coming in and going out all the time. In RACF, if you have users coming in and going out all the time that was a problem because they might own resources used by others. Deleting a user involved reassigning ownership of all the resources that they owned but which were going to survive them. On the other hand creating new objects and destroying objects all the time that was not a problem.

ACF2 came into its own in an environment in which most of the objects that were created were accessed by their creator. So the default was if I create an object, I can access it. I can grant access to other people, but by default, I'm the only one who can access it. And so if I can create and destroy things at will, and I'm not impacting anybody else, and since most of the objects that I'm going to use; and this goes back to the student environment where the emphasis is not sharing, the emphasis is on the student not interfering with other students and not being able to interfere with the system itself. So each of them had its role and its place, and we were lucky that we ended up with all three choices. But you would see very weird things like I went to a CA conference one time where a guy made a presentation and he said he had 800 ACF2 rules. And I thought you have elaborated ACF2 into a list-based system. You probably should have chosen Top Secret or RACF in the first place because maintaining that rule set just became prohibitive; he had a staff of three or four people maintaining the rule set because he had so many rules. There's always the tradeoff in access control against security as opposed to administration. If I have, depending on the environment I'm in, either a permissive set or a restrictive set, may amplify the amount of administrative activity you've got. One must balance the access control policy that one chooses against the amount of administrative activity that it's going to generate.

For example, in a RACF environment, if you have students creating and destroying objects hundreds of times a day, the administrative load of that might be very high. But in a business environment, it's a different tradeoff. Always the problem of trading off administrative activity against security is always going to be an issue. And we see that even in identification and authentication. If you improve the security of the password rule, in theory, you may increase the amount of support that you have to provide in resetting passwords. And that's one of the reasons why in the internet, what you see is mostly self-administered passwords. If I free up my password, we go through a reset procedure that enables me to choose a new password but which doesn't require any administrative activity on the part of the system; on the application owners because we couldn't afford it.

In modern systems the object of control is often the application, the application does its own user identification and authentication, and the user registers herself to the application. This has relatively low administrative load but offers far less granular control than we contemplated when these access control products were developed

Yost: I know that CA took over Top Secret, as they did with ACF2. Do you know the origin and history of Top Secret? Is it a small company forming, like Barry Schrager did with SKK?

Murray: My recollection is that Top Secret was a Computer Associates product. I think that CA offers both products affirms the argument that all three products have a place.

Yost: Also in the 1970s, of course, there were a number of research papers, including those by Bell and LaPadula. Did that model have any impact on your thinking or IBM's thinking with regard to security?

Murray: To the extent that it influenced the TCSEC, it did. It was certainly an important part of the dialog. We were a lot closer to the implementations and product, and everything we thought about was how was this going to impact the legacy products? How is this going to impact the customers? So we didn't have the freedom that they had because they're dealing with abstraction. We were always being guided by customer requirements and retrofit requirements, legacy applications.

If you look at Apple and Microsoft, one finds two completely and totally different philosophies. At some point or another, Steve Jobs would say I'm not going to support that product anymore; I've got a better idea, I'm going to do it better, and I'm just going to abandon all those users and all those applications on those earlier systems. Microsoft never did that and Microsoft would never, ever have done iOS.

Yost: Probably saw a greater financial risk to not have that backward compatibility.

Murray: Right, backward compatibility; all that stuff that's out there. Bill Gates would say, I want a customer to be able to install a new operating system and not break any applications. While there have been lots of iterations of Microsoft where one had to reinstall an application in order to put in a new operating system, the application would run on the new system. I went from Vista to Windows 7 without having to reinstall any applications; and probably wouldn't have gone to Windows 7 if I was going to have to reinstall applications.

I have now reached the point that I've bought the last piece of hardware I plan to buy. I have enough hardware already to interface me to the internet. What I would like to have right now is for somebody to sell me a running image of Windows 8 in the cloud. I could move all my data and applications from my current hardware and never have to have another; never install another piece of hardware or another operating system for the rest of my life.

As it stands, I've got two XP systems there; a Windows 7 system there; an OS-10 system under there; and a Linux system over here. And I operate all of them from that iPad. So to me, all the applications on that system look the same to me on my iPad as they would look if they were in the cloud. It just happens that I own the server that those applications are on and these two systems look to me like file servers [laughs]. I never have to install another operating system or another application.

Today's children are growing up in a world in which one doesn't install operating systems or applications, they just are. I push the icon on the screen and I've got the application running; and if I want to update the operating system it tells me that it's time, I press the button and it says "wait." It takes a while; when I upgraded to iOS-7 it took about an hour. But that's all; you know, it was not a big deal. The interface was a little richer and cleaner but all of my applications continued to function. For a few days, I received a few more application updates than usual. I notice because I do not have automatic updates enabled, but those who do have it enabled would not even notice.

This works in large part because Steve Jobs had the courage to say we are going to have a closed system and we are going to open it up very gradually over time. That was the same philosophy that IBM had with the System 38. They said closed system, RPG only; generated programs only, no procedural programs permitted; no COBOL. Well, over time, they relaxed those restrictions and they opened up the system to more and more things. But they kept the system boundaries in place and you still have to import and export, right? And the only time the AS/400 system is really vulnerable is if somebody can corrupt a copy of the backup and then reinstall it. And they've now even got integrity checks on that so that if the backup gets tampered with you're at least going to know that it got tampered with while it was outside your system.

BLACKER was an early A1 system. It was so tightly controlled that one could not change user credentials while it was up and running. They said not to worry, it crashes two or three times a month anyway, and we change user credentials while it's not working. That violates the continuity of control rules that are supposed to be implicit in an A level system. So always, there are these really, really hard tradeoffs that you have to make and that's in part why we are where we are. We are where we are because we made hard choices all along the way.

Yost: I want to return to something that you talked a bit about and just get some further thoughts from you on the MVS Integrity Commitment that IBM made to its customers in 1973. What do you see as the dominant rationale for doing that? Was it seen at all as an economic risk to make that guarantee?

Murray: Oh yes. The product managers were all uncomfortable about the implications of that. However, McPhee said the quality of the code is good, the procedures that we have in place to produce quality code are good and the risk acceptable. The Commitment can be seen as a way to keep it that way. It has not been without cost but one can argue that it is cheaper than the alternative.

There are still people working on MVS who, when they hear how development is run in other shops, just can't believe it. The rigor and discipline that was brought to that development effort was sufficiently high that the people closest to the product were confident that it wasn't going to be a problem. And it never was. In the first five years or so of the Integrity Commitment, the number of integrity exposures that were detected was small, mostly in old code, mostly device support. Microsoft now argues the same. They assert that most implementation induced vulnerabilities are now in old code that most "blue screens of death" are in device support.

Incidentally, device support is another place where Microsoft differs from Apple. First, Microsoft sells software to hardware vendors who may make changes to Microsoft code to support or distinguish their offering Apple sells a hardware/software package. Microsoft is open to devices. It says write the support for your device according to our API and we will allow/support the attachment of your device. Apple is more closed.

Most of them were not particularly difficult to fix, and could be fixed without breaking applications that were running. I had one, one time, that broke all kinds of stuff but it wasn’t on MVS, it was on document formatting software called the Document Composition Facility. I got a call from John Tabor in Dallas. John wrote a very famous paper on a set of frauds. John says I’m working on the Professional Office System (PROFS system). This was a set of office applications that ran on a mainframe and was installed in the White House. He said we exchange unformatted document attachments. When a user wants to display a document, we format it in real time, using your product, the Document Composition Facility. He said, you have an escape mechanism in that application, called the “.sy” tag. Whatever data follows the dot-sy tag — think like html tagging — is passed through to the environment to be interpreted and processed. So I can send you an unformatted document with a dot-sy tag in it, and some commands for your environment, and when you open that document, just open it — all you did was click on it — that script that I embedded in there will execute in your environment. Sound familiar? Today we would call the document bait, the malware was the hook, and the tag the mechanism for getting the malware executed. When I understood what he told me, I sat down and I wrote a two-line script, which if I embedded it in a document and sent the document to you, and you opened the document, would cause your entire file system to be bundled up and sent to me over the network. In a mainframe system, that operation was so fast that you wouldn’t even notice the one or two second delay.

I was getting ready to push the send button. “God, what power. That’s how the hackers feel. That’s why it’s so addictive.” They’re trying to repeat the rush of the first time they were able to take over somebody else’s system or data. I restrained myself; I didn’t send it to anybody.

The user thought of the document as data. The escape mechanism introduced a means to conceal executable code in the document. The user was not likely to appreciate the danger. Even if he did, we had not provided him with a means to mitigate it.
The fix was to provide a control. In order for the escape mechanism to work, one had to turn it on at object invocation time. That left the question of how to set the default, on or off? If we set it to off, everything's going to continue to operate. If we set it to on, we're going to break those applications using the .sy tag intentionally? For example, the application might be querying the environment to learn something about the file system and the things that are available to it in the file system. It's perfectly legitimate. We set the default to safe. That is, unless the application invoked the dot-sy tag at invocation time, it's not going to work. We broke tens of thousands of applications all over the world. A few months later, I was trying to do something that I had done 100 times and it wouldn't work. It took me 45 minutes to figure out the reason that it wasn't working was the change that I had authorized myself. I broke my own application and I thought, God, if it took me 45 minutes to figure out what was wrong, how disruptive this had been across all users and applications? All of this to fix what was admittedly a gaping vulnerability, with unforeseeable consequences, but no threat. Of course that is the reason that providers so often opt for the unsafe default.

Incidentally, it turned out to be one of the quickest fixes IBM ever made. From the time that John called me until we shipped the change was three weeks. Nothing had ever happened in the IBM Corporation in three weeks; certainly nothing that disruptive. We reacted to the size of the hole rather than to the size of the problem. And if we'd known, if we'd really thought about it thoroughly and run it through the normal management process, it never would've happened; just no way you would've shipped that thing in the safe mode. And again, we make that choice over and over and over again. That's one of the few times we made it in theoretically the safe direction, but at great cost to the customers. And incidentally, when we shipped it, we told them what we had done; we didn't tell them why we had done it and we didn't tell them what the implications were. We just sent it, saying this is a change we made, because we didn't want to expose all systems out there where the change hadn't been put in. And that was generally the case with the MVS integrity fixes, too. We should ship it and we would say this fixes an integrity exposure. We wouldn't say what the exposure was. To this day, when you evaluate CVEs, IBM's problems get a higher CVE rating, mostly because IBM doesn't tell everything it knows about the exposure. And SHARE and Guide both made the determination that's the way they wanted it to work. Most customers, if you had told them what the exposure was, would not implement a workaround.

We now have a minor industry populated by people who style themselves "security researcher" but who are really vulnerability pimps. They expose vulnerabilities and publish exploit code because they think that when they report a vulnerability, the owner of the code should drop everything, including prudence and fix it immediately. Not all vulnerabilities are problems, not all problems are the same size, and sometimes a fix may be worse than the original problem. The discoverer of the vulnerability is rarely in the best position to judge what ought to be done.

Yost: In SHARE and Guide, was there much research done on the economics of security implementation? We talked a lot about tradeoffs, but were there specialists that researched economic impacts of different directions taking different paths?

Murray: No, not really. There still are not. Security people are particularly vulnerable to the appeal of the "safest" course. That is why we stress "risk based" security and the principle of proportionality. Courtney taught us, "Never spend more money mitigating a problem than tolerating it will cost you." The vulnerability pimp is a critic. He does not own the problem and it is not his money.

Yost: I looked through that literature and I haven't found much.

Murray: We didn't understand the importance of what we were doing. We didn't understand how lasting the impact was going to be of the things we were doing and we didn't understand how fast things were growing. We were like the little kid who believes that the world has always been the way that he found it and it's always going to be that way. He can't observe change. And I see it today between myself and young adults. Their world is changing more rapidly than mine did when I was their age. We've gone from the iPhone to the iPad in six years. We've gone from the first iPhone, which did a little web browsing but didn't run any applications, okay, to IOS-7 in that many years. So they can see some change in their lives, but it's difficult to appreciate. That is why we study history.

We didn't know how fast or how large things were going to grow. If you had told me in 1956 when I was working on the 704 that I would carry, would not go outside the house without, a computer in my pocket that was infinitely more powerful than the 704, I could not have conceived it. Suppose someone had told me that there would be hundreds of millions of such computers and that each would have the potential to connect to any of the others, it would have sounded like magic. Or that it would be connected to all the knowledge in the world; everything that is known is somehow or another connected to this tiny computer. It's either already recorded and in the network someplace, or it's in the head of somebody who is attached to the network. So we really already have an artificial intelligence. The internet is really already artificially intelligent.

Von Neumann did not believe in artificial intelligence because his computers were all deterministic. He understood that intelligence requires trial and error, but his machines were deterministic. He said anybody who pretends to compute random numbers is clearly in a state of sin. The computer is deterministic, therefore, it can never be intelligent. But now, I've got social networks, literally billions of people who are connected to the edges of the internet all contributing their knowledge, all contributing randomness, and all contributing error. So we really do have an intelligent machine; could we have conceived that? No. In 1956, 1957. 1958, we did conceive of plastic money. We really didn't believe in it, but we actually have a White Paper or what was called a Blue Sky Paper in those days. We had a Blue Sky Paper that described carrying abstract money in tokens. We really didn't ever think children would use them, but of course, in Hong Kong, you get your Oyster card when you're three or four years old. You have to have it to get to school. [Laughs.]

Yost: John Diebold also wrote some things early on a cashless checkless world.

Murray: Yup. Diebold saw it. Of course, the real visionaries were people like Vannevar Bush, who really conceived of the modern internet, but nobody really believed in it. Who was the guy? Nelson, there was a guy . . .

Yost: Ted Nelson.

Murray: Ted Nelson was one of my heroes. He was an early advocate of computer based education, but the systems that we had at that time were so primitive compared to where we are today that so was the concept.

Have you ever seen Bridger? Are we okay on time?

Yost: Yes, we're okay. Can we take a short break?

Murray: Yes.

[Break in Recording]

Murray: I had forgotten all about Eldon Worley, because when I went back to SHARE to announce the product I was embarrassed by the number of people [who said], isn't this what you told us wasn't a product? [Laughs.] And I said well, it's much more than it was when Eldon presented it to you; we've productized it, packaged it, and put all the bells and whistles and support around it.

Yost: So one thing that you included in your e-mail was the SHARE White Paper. Can you tell me a bit more about it?

Murray: That was an effort that Barry led; Barry Schrager led that effort and like lots of SHARE White Papers, it became a requirement. Once he presented it out of committee to the bigger body, it became a SHARE requirement statement. I think that IBM thought that what we had, RACF, was responsive to it. Barry obviously did not. It was not what he had in his head as the abstraction, and he went back and built what he had asked us for, ACF2, and he made a fortune out of it.

Yost: University computing centers were a relatively small market compared to business users. Was it strange that Barry Schrager was leading this effort with that White Paper?

Murray: Like a lot of other SHARE efforts, these were personal things; there was some personal interest. It was leadership, right? Somebody's providing leadership; and he was providing the leadership. I haven't seen him in a few years; seems to me I saw him at an

RSA conference five or six years ago, but he's more of an entrepreneur today than he is a security person or an IT person.

Yost: Did the access control system that you had built earlier have a significant impact on the first generation RACF, as a product?

Murray: I think Eldon's ideas predominated. I advocated RACF because it was based on working code. My mentor, Joe Wimbrow, under whom I first managed system programming, taught me, "Never commit to a schedule, or even admit to the existence of an idea until you have code that runs." Based on that lesson, RACF was the least risky of the courses open to us.

Yost: [Elson] came to Poughkeepsie for six months.

Murray: Yes. I think his ideas and his code, probably, predominated. And RACF had a lot of development done; you know, there were requirement statements against RACF almost as soon as it hit the marketplace and it had continuing development. I don't know what happened in the last 10 years or so, but for most of its life, it was continually being updated and some very interesting things were being put into it.

The concept of privilege in RACF, I think is unique to this day. While most systems have one all-powerful user, RACF has three users who share the privileges in an interesting way. There is the operator, the auditor, and there was the security administrator. Each of them has a role, things that they could do that the others could not do. There was no concept of "root;" of an all-privileged user in RACF. The operator has global access to data but cannot change the security rules or control the logs. The security administrator controls the access rules but cannot access any data. The auditor controls logging and logs; he can hold the other two accountable for their use of their privileges but he cannot access the data or change the rules. Each, simply by carrying out his assigned role acts as a check upon, and is checked by, the other.

I once had an online debate with Dorothy Denning over the sharing of user IDs; she insisted that it was often necessary while I argued that it violated strict accountability. What really needed to be shared was not the identity but the privileges bound to it. To address this requirement, RACF implemented surrogacy. A, for example a manager or executive, could authorize B, for example a staff member, to exercise their privileges. B logs on as A, with all of A's privileges; the audit trail would show that that's what was happening. All subsequent action would be logged as "B acting as A." While acting as A, B cannot use their own privileges; thus, both accountability and separation of duties are maintained.

One of the places that the surrogacy got used was to empower operators when the principal couldn't be there. A might be the application manager, and the application crashes in the middle of the night. B, the operator, calls A and asks what do you want me to do? A says, do thus-and-so, thus-and-so, thus-and-so; B says I don't have privileges to do that. So A grants him surrogacy for the period of whatever, and he can make the necessary changes that A wants made.

There are all kinds of really, really neat controls that are built into RACF that simply haven't shown up in other access control facilities, many of those going way beyond what Eldon conceived of, but which were added in response to the increases in scale, as well as the articulated requirements of customers.

Yost: Were SHARE and Guide meetings critical to deciding elements of subsequent iterations of RACF?

Murray: Yes, very much so.

Yost: Do you recall the rationale for the early pricing strategy for RACF?

Murray: I don't think so. Nothing comes to mind except at that point, software had only been priced separately for five, six years and it was not well understood. I don't think that the price was ever an inhibitor to its adoption. However, I have always admired Netscape for making [it] a standard feature. I suspected that IBM would have made it a priced feature. Netscape treated it like "brakes," essential to success, while IBM might have treated it as a "nice to have" option.

Yost: It was the resources that the customer invested to operate [pause]

Murray: Yes, it was the customer resource; and even the customer understanding. We knew that adoption in the U.K. for example, was extraordinarily high very early. It was because we had a data security rep in the U.K. who was very articulate and was able to better help the customer understand the value that he was getting. We couldn't replicate him, but we did use him to try and instruct the other data security reps. I think at one point we had about 15 or 20 of them around the world. However, their role was to support the customer, not take orders.

Later on, RACF got its own marketing effort, but in the early days, what marketing effort there was on RACF was coming from the data security reps. As I say, we had about 15 or 20 of them around the world and we would bring them in periodically for training. And they would consult with customers. I'm trying to remember the name of the guy who was in the U.K.; I can see his face but I can't remember his name.

Becky Bace worked with [Bob Abbott], so she should remember a lot of that stuff that Bob would've remembered. Again, the RISOS effort is — and Matt Bishop will tell you that too — it's an incredibly important piece of work that is not taught in the colleges and universities. The people who are teaching programming are not disciplined people; they are all; and they've never worked in a disciplined programming environment; they've never worked in a big development shop where quality is being controlled. They look on coding as a game. There are some software engineering shops where structured programming is being taught, but there's nothing the equivalent of 'strength of materials' being taught anyplace in colleges or universities. There's nothing about how to avoid; nothing like Lipner's program that he's got at Microsoft, where the code is vetted against certain common kinds of attacks, for example. Nothing like his shop is being taught in the colleges and universities, and that's another source of our problem. Programming is being taught as an art, not as a discipline. Even the people who talk about software engineering don't adhere to the traditional engineering quality models. We have Easter eggs instead of people signing their work, right? Even the movie industry does a much better job of producing movies than we do of producing code. One of the best project control shops in the world is at Disney and they teach project control, and they are very good at it because that's how you get a movie out. I remember riding up the ski lift one day with a textbook publisher and he just couldn't get it through his head. He said I've either got people who know the technology or people who know the subject but I don't have anybody that knows both. And I say, that's not how we produce movies; read the credits on the movie. Even the kid who carries the lunch got a credit. [Laughs.] And that's how we're going to have to produce textbooks in the future, right? Now we see the Apple textbook effort for iPad and we're going to have production shops, we're going to have Pixars producing textbooks because it's really, really, really important.

Yost: You brought in an essay or article that when you were first managing, or early on in managing programming a project that . . .

Murray: Real programmers can't do it.

Yost: . . . that real programmers were more of a problem than new hires that didn't have training.

Murray: Right. Exactly. To this day, I still believe it's true, you know? If I had a big project to do and quality was really important, I'd train my own from scratch; I'd give them the right tools; I'd give them the right workbench. And, you know, "Real programmers" the myth that we can produce programs without rigor and discipline faster than we can do it with rigor and discipline. And I've seen this over and over again, even in consulting practice. I've got a quality reproducibility metric for consulting that says that one ought to be able to give a consulting effort to two completely different teams and get the same result. Never gonna happen. The result they produce ought to be replicable, but rarely is. It's because each of them brings his own absence of method to the project. If one tries to give consultants methods, procedures, or tool books, they say we don't have time to learn it. Frequently they don't because the suggested tools and methods they're given are too complex and require too much learning time in order to be able to apply them and they don't have learning time within the project to do that. So they learn that everything is produced *ad hoc* and each programming or consulting effort is completely and totally different from every other one that they've ever worked on.
We have the situation in the world where most applications are built by people who are working on the first application they've ever done; we're not accumulating the experience in application development shops that leads to reproducibility and high quality. It's sad, but it's true; we're not getting any better. We are not maturing as a discipline.

Compare to engineering. Engineering as a discipline is 6,000 years old and all the experience acquired over that 6,000 years is being recapitulated in the education of new engineers. They know how to build for the ages; they know how to build infrastructure. Infrastructure should be built like Boulder Dam, right? It should not fall down in a thousand years; and it should be maintainable and operable, but it should stand forever and ever. We're not building computer code like that; we're not building code that we expect will last a long time. The security subsystem for AAS lasted 15 years. We didn't build it for 15 years; we built the whole system, I think, to stand; we built to infrastructure standards because we had some sense of what we were doing.

We had the experience of Sabre; Sabre's still running. The earliest big online real time application in the history of the world is still running and a lot of the original design decisions are still in it. Sabre ran with code modules that were no more than 1,000-something in length, the length of a track on the disk drive. The reason was you had to be able to retrieve them in one rotation of the drive. [Laughs.] And that was how long the track was so we had to fit it on the track. Later on, we built solid state memories to hold the programs so that they could be retrieved at random at high speed. But the program length never changed; it stayed the same. I had a colleague who left IBM because we had a constraint on AAS that says a module might not exceed 4K. He said, "You don't understand, I'm building accounts receivable application and we have to have modules that are bigger than that. If you impose that constraint, the system is bound to fail and I'm going to leave; I'm going to go to New York University and teach 'real programming'." And that's what he did.

It turns out that if you look at the code, 90 percent of the modules didn't exceed 2K; never got close to 4K; didn't need to. The idea that I can take something and break it down into single lines of code, but there is no module between this and one line of code, no modularity that would allow me to break it up into smaller pieces. I said that's absurd on its face, but we had all kinds of people out there who still think that way. When I look at the DLLs that are in modern Windows, I can't understand why they have to be as big as they are, and I know there are all kinds of problems in that code because those modules are too big; they're too big for anybody to comprehend. I don't understand why all that code's in there. Part of it may be that they are putting data in the module, but I don't want the data in there either. I don't want to mix up code and data.

Yost: You mentioned TCSEC earlier. Was the group at NSA in close contact with IBM or with you personally getting advice on industry perspective?

Murray: Well, it was something of an adversarial relationship because they were dealing in broad abstracts and we wanted something that was implementable. It turned out that the authors of the TCSEC thought that one would take an arbitrary product and determine which of the four categories it fell in after you held it in your hands. It turns out that you had to have the category in mind when you designed the product. The category that something fit in was not an evaluation, it was a design point. I don't think the original authors had that in mind. Nor did they have in mind the methodology for doing the evaluation and as a consequence, they had no idea how expensive an evaluation was going to be; they had no idea how long it was going to take; and how that was going to compare to the product life. We were doing essentially disposable products. The product life of a version was measured in months to a short number of years, and the evaluation process was taking that long. They eventually came up with a concept they called RAMP, which allowed the owners of a product to make modifications that would not affect the evaluation.

Then there was the problem of comparability of products within the same evaluation. There were those who said C2 is C2 is C2. And I'd say no, MVS is evaluated at C2, and Microsoft DOS is evaluated as a C2. Those two operating systems are simply not comparable in functionality or scale but from a security perspective, they both fit the same set of criteria. The cost of evaluating one cannot be compared to the cost of evaluating the other. Some of the cost of the evaluation was absorbed by the government, who is paying an independent evaluator. But the liaison required from the developer to the evaluator was very, very high; it was a very costly process.

Microsoft found out the same thing when they tried to submit things for evaluation — by the time the evaluation was done, the product had essentially moved on to new environments and new criteria. The evaluation of Windows NT took more than a year. It was only reliable if the system was unconnected. All that expense was for a configuration in which few systems were ever operated.

Courtney used to contend that the government spent literally billions of dollars on third party evaluation as a concept. Courtney said it never made any difference in the security of a single government installation anywhere in the world. Probably too broad a statement to be literally true but “close enough for government work.”

On the other hand, I think the lists that appeared in the TCSEC, the criteria, were useful; the list of all the things that must be considered. For example, don’t forget to consider covert channels. It may or may not be applicable to your product but you at least need to know that that kind of an attack exists, and you have to have thought about what it means.

One of the things that happened at IBM was that we came up with the concept that every product manager has to consider security. The policy for all products was that they could be operated safely for the intended application and environment. After that, he can say pretty much anything he wants to about it but he may not be silent. He may say these are the features, functions, and properties that I have provided; this is how I expect the customer to use them; the ultimate responsibility for security rests with the customer; but he may not be silent. He has to say something on it no matter what. Testing of his language built into the product announcement review cycle. I think that the fact that he could not remain silent motivated him to build in the necessary features, functions, and properties. However, over time, the language became less descriptive and instructive; degenerated into boilerplate. Product managers got to the point that they would say "security is the responsibility of the customer." But for a long time it was very effective control; it really was making a difference in the products that were produced.

The product managers were told that the product has to be safe for the intended application and environment. While that may be very different from anything that's in the TCSEC it generally worked. Most of these were not "shared resource computing systems;" the applicable criteria was "D" People think that "D" is a "bad grade," that if a product is a D, it isn't any good at all. Not true. The D evaluations are some of the most important ones we have. We use D, for example, for chips and cards. We use it for encryption engines. We use it for all kinds of stuff, and it's very, very important and valuable. We use it for things that will be produced in large quantities and for which we will rely heavily. The letter refers to the criteria, it is a category, class, or set of requirements, not a grade. Do not rely upon it as a grade; read the evaluation report. Note that while the authors of the TCSEC asserted that A is good security, and that B is the "class of interest," we hardly ever use A or B, in par they were intended to address problems that have all but disappeared. They were designed to solve the problem of sharing expensive hardware. Hardware's not expensive anymore. Even at the time, I kept telling them to use more hardware and rely less on software. If I've got two different physical boxes and I can watch the channel, and the two boxes are not costing me that much more than one big box, I'm better off than if I've got everything in one box. But ultimately, even if you're not sharing hardware you're still sharing data, and we're still relatively primitive at controlling the sharing of data. The most sensitive data that most organizations have is in plain text and we don't have document management systems in place. We don’t treat the life cycle of a document that’s documenting our intellectual property the same way we would treat the life cycle of books of account but we need to do that. One of the things that’s happening right now is the way businesses operate, they get sued. And when they get sued they are served with a discovery order, and they don’t know how to respond. Their lawyers go out and call in people who are now called forensics experts. However what they really do is go in and do a document discovery procedure. Where they identify all the information that’s in the organization and index it. That’s step one and it’s independent of what’s going to happen after that. Step two says now that I’ve got this huge index I’m going to go look for evidence and then when I find evidence, now I’ve got to turn around and fix that evidence so it can be presented to a jury. And the people who do that work; the guy who does the first step, who just discovers what you’ve got and index it all, he’s going to charge you millions of dollars.

Yost: It’s a huge effort.

Murray: It’s a huge effort and it’s much more costly than if you’d had the document management system in place all along. And they would’ve enjoyed all other kinds of security advantages.

The reason they assert for not having document level control is that it’s too big an undertaking. I say, start with your most valuable intellectual property, put it on a Lotus notes server. Often they can’t even tell you what the most valuable intellectual property is. [Laughs.] The Chinese, who are already into their system, know more about their intellectual property than they do. That's scary. But, you know, getting control of your intellectual property can be a very, very expensive process if you haven't been doing it all along.

Yost: In 1984, in the *IBM Systems Journal*, you published *Security Considerations for Personal Computers*. Can you tell me about the context for writing that article and the responses to it, the impact it had?

Murray: It was a very early effort, you know. At that point, we're three or four years into personal computers. It's sort of like discussion around cloud security today. No, better yet, mobile computing today, okay? It is emergent, everybody is beginning to understand it's important and there's going to be sensitive data there, but they have no real sense of how to address it. What little they know comes from the past, and is technology dependent. There is a desire to use the tools we know control it the same way that we did the previous. We controlled mainframes at the hardware layer, so when we got ready to try and do PCs we wanted to do the same thing. We didn't understand that the management structure around that PC was very, very different; we wanted a hierarchical management system. However, the earlier technology of interest was not the mainframe but paper and pencil, users rather than machines.

One of the things that got IBM into business trouble was the failure to recognize when the scale of computing changed the scale of decision making. One of my colleagues, Dale Weingarten, says his career went downhill from the day that he walked into Frank Cary's office and demonstrated a PDP. Cary could not understand that that thing was ultimately going to compete with his business. And when we first started to sell PCs, we, and the CIOs that we thought of as our customers, still wanted centralized control. We didn't realize that the decision about who was going to buy something was no longer being made by the chief information officer; because the scale of decision has changed so has the locus of the decision. As the scale changes, so does the locus of security control. When I first started to sell computers, even chief executive officers did not have the discretion to buy a computer. The decision to buy a computer was comparable to that to undertake a new product or build a new plant. The investors, the board had to make that decision, right? Couldn't be made by the CEO. After it became more routine, and the prices started to come down, then IBM invented the chief information technology office so they'd have somebody to sell to. Somebody who had sufficient bandwidth to appreciate the product and sufficient authority to buy it. We had to do the same thing when the scale of printers changed. When IBM built the 3800 printer, the big thing that was two-thirds the width of this room. No single person in the business could make the tradeoffs that were involved in replacing all those little printers with that big printer. And so they had to invent the surrogate to make that decision.

Today, we have — I was talking about it last week at an ISSA meeting — we have this entity in business called the risk manager, or even the chief risk officer. He was invented by the insurance industry to buy insurance, so they'd have somebody to sell to; somebody who was an employee of the business who could go to the upper management and to the board and make the case. Someone to say,"this is an insurable risk and it's going to cost us this much, and we think this is the right carrier for the insurance, and all that sort of stuff."

So as scales change, decision making changes. I'm convinced that today; well, I've just had this experience. I was chatting on IM with my niece, who is 18, and I said Sydney, do you have a birthday list? She says yes, an iPhone 5S. I said which model? She says gold. I said how much storage? She says 32 gig. [Laughs.] The decisions are being made by the children. Decisions in business are always made one level below the guy who signs the bill. So today, the kids are making the decision about which hand held, mobile computer am I going to buy? And the parents just pay, and then they review the recommendation, right? I did ask, why 32 gig and not 16? She says, because I've already had to offload some of my photos from 16 gig and I'm bumping up against the 16 limit. So I need to go to 32. Okay, I said, I understand that. The scale changes and so the management processes have to change. So I was trying to identify management processes that scaled to the device and again, that focused on the data; not on the technology, and not on the hardware. Be independent of the technology and the hardware.

Yost: Can you tell me about your decision to retire from IBM and join the consulting firm, Ernst & Young?

Murray: Yes. On a Friday, an article appeared in the Wall Street Journal that said IBM is going to offer some middle managers an early retirement option. I was then 51 and was planning to retire at 55, in any case. And on Saturday, my phone started to ring and I began to realize that I enjoyed more reputation outside the business than I did inside. One of those phone calls came from David Wilson, of Ernst & Young, who was then working on the Clark Wilson paper. We had started working together when he was at Monsanto, and was very interested in computer security and attended a lot of my programs. He had gone on to Ernst & Young from Monsanto, and was building a security consulting practice. So he says come to work for me and I did. We had a very successful relationship, built the practice up considerably, and then came the merger with Arthur Young.

Yost: This was 1986 that you started at Ernst?

Murray: 1986-89 I was there. When the merger came, the business got out of control. What they really needed was a consultant to come in and help them reorganize, but they did it from the inside and their decision was to "go back to basics." Go back to taxes, and audit, and accounting because we do that really very well; we're going to strip off all this consulting and security stuff. And so I went with Deloitte & Touche with Ed Johnson. At that point, Dave Wilson, my champion inside the firm had moved on. It turns out that he was such a good executive that they couldn't afford him on this practice, they needed him to take up broader responsibilities. One of the first things he did was work in the office of the chairman for a couple of years as an executive assistant to the chairman and so my champion inside Ernst & Young went away.

Deloitte & Touche was also merging, but they were growing in the process of their merger. Part of Deloitte & Touche was Touche Ross, and Touche Ross had always had a premiere forensic practice. They were the people who went in after the disasters and cleaned things up. Their forensic practice cut their teeth on a bankruptcy called Equity Funding, an insurance company that engaged in a massive management fraud not to be equaled again until Enron. Touche Ross were the forensic auditors that went in to clean that situation up. In the merger, Deloitte acquired Touche Ross. They also had some good security leadership already in the firm. Everett Johnson was the partner who was leading that practice. Ernst & Young had been in Cleveland; Deloitte & Touche was in Wilton, the next town over from mine. Harry DeMaio, who had led the corporate security function at IBM, had retired about a year before I did, was also at Deloitte. Harry likes to say we retired before it was either necessary or popular. The deal was sufficient. Bob Courtney had retired under a similar deal. When somebody asked him why he had done it and he said "the money." The money made the decision easy but it was really the opportunity to do fun stuff, fun work, work for which I was better qualified than most.

Yost: Were you doing a lot of forensic type work?

Murray: I have never done a lot of forensic work because my contention is that I lack an essential kind of experience, that is, I have never worked under investigators. I have worked under auditors; I can do forensic auditing. But I can't do legal forensic investigation because I've never worked under those kinds of investigators. Now, I teach forensics, and I teach computer evidence, but I'm not competent to lead a criminal investigation.

Yost: So was your consulting more developing policies and procedures for security and auditing?

Murray: Architecture, governance, policy, and applications. And I had some really fun ones. I worked on architecture engagements at Coca Cola, British Airways, Foxmeyer, and Wells Fargo among others. I did an architectural methodology with Deloitte & Touche. Never got taught and conveyed but it's a good methodology. We did get into the practice a lot of architectural principles that kept us out of a lot of trouble. Like one of the things was to be sure to use products as they are intended to be used; don't stretch the envelope. Don't break interfaces. That is, don't exploit what you know about what's behind the interface. When you use the interface, use the interface exactly the way it was designed to be used. Don't bend things to fit. You may build a shim, that's okay, but understand that when you build the shim there are going to be constraints that apply to how you do the shim. So a lot of that stuff got into the culture that I think has been very beneficial to their practice.

Yost: Can you talk a bit about Certification of Professionals, the Common Body of Knowledge comes in 1990, and then the CISSP in 1994?

Murray: At some point in the 1980s, I was at an EDP Auditors Conference. That was before EDPAA had become ISACA. At a cocktail party one night, with most of my peers and colleagues in the room, I suggested that we should form a college to credential ourselves. One of my senior colleagues, Marty Silverman, threw the Groucho Marx line on it that said I wouldn't be a member of any club that would have me. [Laughs.] And so the idea of the college went away.

Nonetheless, we were experiencing the same problems that have confronted every emerging profession including how to separate the professionals from the amateurs. It was particularly important for us because of the amateur, i.e., "hacker," culture that surrounded so much of what we do. Some kind of recognition scheme was needed.
A few years go by and the idea of professional certification re-arises. Under the leadership of Michael Corby and others a group of institutions, each of which thought they had some claim to authority to certify security professionals, got together and agreed to form a consortium to certify professionals in the systems security space. Under the agreement surrendered any claim to authority to this organization, which was to become the International Information System Security Certification Consortium, $(ISC)^2$. In return each organization got a seat on an advisory board to be able to govern the integrity of the program and its certifications.

$(ISC)^2$ would go on to become the governing body for the profession with more than on hundred thousand certified professionals world-wide. The Certified Information System

Security Professional (CISSP) credential would become the "Gold Standard," the most coveted credential in information technology.

The board floundered for a number of years, trying to bootstrap this idea into a sustainable business model. It turns out that writing an examination is a difficult and expensive thing to do.

One of the contributing efforts was the ISSA committee to describe the common body of knowledge. The existence of such knowledge shared by the professionals is a prerequisite for the existence of the profession. Members (Robert H. Courtney, Jr., Harry B. DeMaio, Toni B. Fish, Jerome M. Lobel, Sandra M. Lambert, William H. Murray (Chair), Martin Silverman, Dennis Steinauer, Harold Tipton) of the committee had at least ten years of experience and some more than twenty. We wanted to describe, not recapitulate, but describe the body of knowledge that distinguishes us. We said that body of knowledge is that knowledge that we expect of one another as security professionals, but that we would not expect of our management peers, or our IT peers. But what is it that I expect of my security colleagues? How would I recognize that somebody is my colleague? That's what we wanted to describe and I can remember riding to the first meeting with Toni Fish and we agreed that access control was clearly on the list, that there were things that would fall under that rubric. Backup and recovery were clearly on the list; contingency planning was clearly on the list. The question was is cryptography on the list? We agreed it was.

By the time the committee had met, we had 17 items. We ultimately had to squeeze that down, but for each one of those items we said what is it that we expect people to know about this thing? And that became the common body of knowledge. The framework survives today.

We said what is it we expect people to know and give an example of that. What would an example look like? So it was the base technology area that we wanted to look at, describe what is included in that and then give examples. And even give an example of a chess question that might appear on the test. So that work had already been done by ISSA, and ISSA contributed that to the consortium when it was formed. But getting from there to an exam, and a business model, turned out to be very difficult.

I served on the board quite early for a short time, and then went off. But I was working on the committee that wrote the code of ethics and the ethical guidance, and I was invited to New Orleans to present that at a board meeting. While I was there, Jim Duffy, who was then chairman of the board, was trying to fill a vacancy on the board and my presentation got me that vacancy on the board. My biggest contribution after that was getting the consortium to hire Duffy as our first professional manager. Duffy provided most of the leadership that got the consortium off the ground and turned it into a viable business proposition. There's independence in the exam, there's independence in ethics, the board has now moved from a volunteer management organization to a true governing body, and the business now employs about 50 or 60 people.

"Professional" is what we share with other professions. We do professional work, work requiring intellectual attainment and discipline. We serve principals who could not do our job. The patient who hires the physician can't do the physician's job. By definition, we repose a special trust in professionals and expect special duties of them. Professionals are not supervised in the same way as others; they have to be able to work with more independence. . professionalism is what we share with other professionals. The body of knowledge distinguishes us from other professionals; it's what makes us "security" professionals.

Yost: Did it take a while for industry to come to recognize the CISSP credential and when did that start to occur?

Murray: Yes. And it didn't occur until the late 1990s. We started the effort in the early 1990s and it was the late 1990s when it was completed. One of the things that helped us along the way is that ANSI came along and said we're going to certify certifying bodies. At that point, we had really put our mechanism in place; we were the first people to be certified by ANSI. They learned that we had learned. (ISC)^2 literally became the gold standard, the standard against which all other certification programs are measured.

One other thing that enabled us to bootstrap ourselves was that we entered into partnership with two other organizations, one that did training and one that did psychometrics. Our early revenues were from the security training side rather than from certification.

One of the board's objectives was to make certification a self-sustaining business. Schroeder Measurement Technologies, who were in the testing business, helped us get our testing program up and running. They had the resources to vet, standardize and administer our tests, to administer the process of applications and all the rest of that stuff. So unlike some other organizations, the consortium didn't have any parent that provided it with capital and so bootstrapping it was a big effort. We bootstrapped it, in part, on the capital investment of those two partners. That enabled us to get the business up and running. By the time we got it up and running there was already significant demand and it didn't take very long.

I can remember then chairman, Jim Wade told Duffy, in three years I want you to have 15,000 people to be certified and I want certification to be self-sustaining. And I said you've got to be really careful what you tell Duffy because he's going to do whatever you tell him to do. And sure enough, he did it. [Laughs.] Today, almost anybody who works in IT wants to have a CISSP, in large part because it's not simply the security certification, but it's a management discipline certification. So it says you've got to know a lot about IT in order to pass the exam. You've got to be disciplined and rigorous and you've got to know a lot about management to pass the exam, and you've got to know a lot about IT process, like backup. That's a process, an essential IT process; and contingency planning.

It's interesting, in part, because business didn't do contingency planning before the computer came along. The computer was the resource because in the early days, everybody had only one computer, scaled to the business. The business was dependent on it and it represented a single point of failure. People began doing contingency planning around the computer. Today, contingency planning is a business process. It's not just an IT process. We learned very quickly that we had to plan around the business applications, not the computer to hold up the applications. It was the people; it was the space; all of those things were going to be critical to your using the computer.

But in those days, all computers were built to order. You walked into an IBM plant and over every computer hung the name of the customer for whom it was being built. No speculation, no inventory, no middlemen, no secondary market even, in the early days. When one placed an order, the delivery schedule was a function of the time required to build the computer.

As volumes increased, schedules shortened, and scale decreased, contingency planning turned into a business process. Well, who knows the most about the business process of contingency planning? It has to be the security people. So the certification is very highly valued. And then the government said we want it. And then IBM said well, it helps us in our consulting practice if our consultants have a CISSP. And Deloitte says we want our security consultant staff to have CISSP. And Ernst & Young says we want ours to have it. So it mushroomed very, very rapidly and I think that they're now up to about 100,000 CISSPs. It's a well-run, well-governed business.

Yost: Is there an education requirement, like a bachelor's?

Murray: Yes, and that's the professional side. We test for the body of knowledge. We rely upon experience and education for the professional side. So you're expected to have a bachelor's or equivalent; you're expected to have five years of experience — management experience — five years of "professional experience." The way they evaluate the professionalism of your experience is by looking at the verbs in your description of your experience. Verbs like developed, managed, communicated, those kinds of professional verbs are the way that they evaluate the experience and say this is administrative experience and this is professional experience; this is technical experience and this is professional experience.

Originally, the exam only tested for knowledge and the other side was done primarily by other qualifications. But now they are putting in questions that test for skills and abilities. Can you do the following things? Can I give you a list of things and can you sort those things into threats, attacks, assets, and vulnerabilities? Can you sort them as to good or bad? Those kinds of tests that are going in there. I would argue that anybody who cannot use the terms threat, attack, vulnerability, risk, consequences; anybody who can't use those terms consistently and exclusively — this is, never use one when you really intend the other — is a pretender; they're not really security people. So you can test for that, right? You can test for certain kinds of attitudes, as well. But the test, originally; what we're really trying to do is recognize one another. And the way we recognize one another is we use the same language and we use it in the same way; we share the body of knowledge. We tested for habitual knowledge. It was hard to fake because little of it was written down.

Today, people actually work to qualify; they're trying to accumulate the necessary experience and the necessary knowledge to qualify for this certificate that they know that they want to have. It's like people work to become a CPA. I mean, they go to college starting out, I want to be a CPA. Well now, people can go to college and say I want to be a CISSP.

Yost: Is prior conviction for a computer crime an exclusion?

Murray: Yes, it's interesting how we test for that. We've got a number of questions on the application about background. Have you ever been arrested? Have you been convicted? Have you been convicted of a felony? Checkbox; but when you check one of those boxes the staff then goes back and begins to ask you questions about that. The attorney will go to the police department and get your rap sheet, and decide, based on those answers, whether or not you can sit for the examination. However, we rely on what you say.

Now, one of the obvious things is that if I've been convicted of a felony, I just don't check the box, right? One of the reasons that we put the question on there is so that if later on we discover that you have been convicted of a felony, we don't have to go through any complicated procedure to debar you, to defrock you; you lied on the application, that's sufficient. We don't have to look at what you did or any of those other things.

It's turned out to be very effective. It's amazing the things that people disclose. It turns out that lots of students; there's a lot of juvenile drug arrests, you know? If you've got a juvenile drug arrest but you've been clean for ten years, we don't consider that a bar. If you had it last month, that might be a bar. And so it's turned out to be a very effective process.

We also have a short list of notorious people, people that are known to have been convicted felons in our space. Particularly, any wrongdoing that's in our space; fraud, forgery, any of those things are almost automatic bars. But stuff that's not in our space, stuff that might not reflect upon professional character, won't stand as a bar.

It turns out that the things that we most have to discipline people for has been forgery or plagiarism. Plagiarism has been a big problem in part because when we first started, the common body of knowledge wasn't well documented; people carried it around in their heads. In the first decade after the ISSA report, all kinds of books were written. But when we first started, the IBM Data Security Controls and Procedures that I wrote was the authority for many, not to say most, of the questions. There has to be an authority for the question and its answer. One can't just write down a question and assert an opinion as the right answer for the question. There must be some kind of authoritative source, and that source has to be available for the people who want to sit for the examination. When we started out, the intersection between the documentation and the knowledge was very sparse, and it was only in that space that you can write questions. Within that first decade after the common body of knowledge was published there was an explosion in the number of books. We prefer that the authority be a standard, or that it be from a refereed document, or a document that is otherwise accepted by the community as authoritative. Kahn's *The Codebreakers* qualifies because it cites its sources. The TCSEC is accepted as authoritative because the government cites it.

The questions are vetted first by a committee against rigorous criteria for purpose. The final vetting is by the psychometricians. The role of the psychometricians is to exclude clues for the "test wise."

The experience level of the qualifiers of the exam has stayed pretty constant throughout the life of the examination. The price of it is high enough to exclude the frivolous. (ISC)^2 asserts that this is a certification for lifetime professionals in the field. If one does not intend that, expect that, maybe what you want to do is get a Microsoft certificate, or a Cisco certificate, or something like that; something that is directly work related as opposed to a professional career. But the program has been very successful and the quality has been maintained very high. The ANSI auditors come in every three years, and of course, the business auditors come in Most of us who have been associated with it have been very proud of it. Two years ago, I guess, we had the 20th year celebration, where all the originators and all the board members through the history all came in.

At the time that I retired from the board, it was very professional, a well-functioning board. We had a number of consultants come in and look at the governance process that we had in place, and we scored very high both in the evaluation of the consultant and our evaluation of ourselves against the criteria that the consultant presented to us. So it's something to be proud of. I'm going off to Chicago tomorrow for the third conference — which is a joint conference between ISC^2 and ASIS, the American Society of Industrial Security; it's a huge conference—to talk about ethics, professional ethics. The experience so far has been pretty good; we have not had instances of nonprofessional behavior but Edward Snowden is a CISSP, and . . . [pause] I don't guess Bradley Manning is a CISSP. Snowden may be the highest profile CISSP who's gotten in trouble. I think that without compromising my ability to judge the situation later on, should it ever come before the committee, I can say that there's a *prima facie* case that can be made that Snowden violated all four of the canons. Now, he appeals to a higher set of values to justify that behavior. That's where all the questions lie. But I will point out to you that so do Directors Alexander, Brennan, and Clapper and Alexander appeal to higher value for lying to congress, for corrupting American business, and systematically deceiving the American people about the scope and nature of what they were doing. They say all that's justified by national security. One might argue that national security is subservient to the Constitution; that they didn't take an oath to national security, they took an oath to the Constitution. But the whole area of the ethics of the intelligence community is "special." I worked on a little group that we called the Galway group. We had either a security background or an intelligence background, some ethics background, and the issue was to try to describe a set of ethical guidance for intelligence officers. I finally gave up; I walked away from it. The intelligence people stuck with it but I think what they ended up with is vapid. The reason that I finally walked away from it is intelligence is an essentially corrupting activity.

I have a goddaughter [points to photo], Nicole, graduated *cum laude* from the Georgetown School of Foreign Service. Really bright, great at puzzles, would fit in really well at Fort Meade. I thought seriously about getting them to recruit her but she also has an incredible sense of justice. When she was this high, she would say, “that’s not fair.” She was in preschool, preschool couldn’t start until she got there because she was the mediator of all the little disputes. I figured, you know, at the end of five years at the fort, she was going to come up against one of the ethical dilemmas that confront all of the people out there, and she was going to crash and burn. I didn’t want her to be five years into her career, and crash and burn over ethical issues. Everybody out there confronts them. The secrecy is essentially corrupt and deceit is systematized. [Laughs.] So she would fit in as far as skills and abilities were concerned, but she would founder over the ethical stuff.

Oh, incidentally, one of my other contributions was that I did write the code of ethics for ISC^2 and most of the ethical guidance that supports it. I have a belief that good ethical guidance says do this, don’t do that; and I’ve got some good examples of people who wrote that way. We never would’ve gotten the “don’t do that” stuff adopted by a community that includes a lot of people who think of themselves as hackers. So the code that we eventually wrote says “do good things.” It’s “be-good” ethics, but we supported it with guidance. The guidance says do this, don’t do that. You don’t have to subscribe to the guidance; you have to subscribe to the code and you have to read the guidance; but you don’t have to subscribe to it.

It turns out that most of the ethical dilemmas that confront security professionals are of the form the-lesser-of-evils or the-greater-of-goods. They are not distinctions between good and bad; good and evil. We don’t get confronted with those. Snowden, says it’s the greater of goods. He says the greater of good comes down on the side of transparency and accountability. He claims that trumps his responsibility to keep his client’s information confidential.

Those are the kinds of things that confront all professionals, not just security professionals. It’s hardly ever a choice between black and white. It’s almost always a choice between the greater of goods and the lesser of evils, and so that’s what the guidance tries to do, tries to help sort that out. Another thing that the canon does is establish a hierarchy of “goods.” It says that the duty is first to the good of society, to the community; then to your principals; and then to your colleagues; and all four of these come ahead of your responsibility to yourself.

When you can’t be true to the greater good while also being true to your principals or your colleagues you have to walk away. We can’t say that over and over too many times, so I’m going out to Chicago to say it again on Thursday.

Yost: Finally, before we conclude, are there any topics that I haven't asked you about or things you'd like to say?

Murray: Let me check my list and see what I have thought about and haven't yet talked about. I haven't talked about my heroes: Robert H. Courtney, Jr, Willis Ware, Horst Feistel, Ruth Davis, Harry Daniels, Hal Tipton, Barry Schrager, Jim Duffy, Jim Bidzos, Ron Rivest, Jim Canavino, Whitfield Diffie, Donn Parker, Marcus Ranum, and Ken Weiss.

Yost: What about Martin Hellman?

Murray: I did not know Hellman personally. In these graduate student/faculty collaborations it's always hard to know in retrospect who made the greater contribution. We will probably never know whether Diffie-Hellman was mostly Diffie or mostly Hellman. My sense is it was mostly Diffie, but Diffie's influence on the community has been consistent and great across the generations. And Hellman has not; he's been mostly an academic; he's really not been part of the community but Diffie has been part of the community. So when I look at my heroes; here's the great men list, I think. Courtney, Willis Ware, Feistel, Ruth Davis, Harry Daniels. We didn't mention but Harry Daniels was the guy at NSA who ultimately became the champion of DES within the agency. Without him, the agency would never ever have gone along with it, but he made it obvious to the agency that it just had to have it. He then later on said it was the biggest damn fool mistake he ever made, so he lived to regret it, but he was the champion of it that made it happen. Barry Schrager, both for this entrepreneurial contribution to the industry, but also his technical contribution to the SHARE paper. Marcus Ranum, I guess, is really second generation computer security but his contribution to firewalls and that particular implementation — of course, Steve Walker had a great deal to do with that also, and so did Lipner, because Lipner was in business with them for a few years — but his contribution has been dramatic and his thinking is really very good.

Fred Cohen doesn't make my list of heroes because Fred Cohen is what I think of as a vulnerability pimp; but he is a good friend. [Laughs.] He did a presentation at IBM Research that I was invited to while I was at Deloitte. He was talking about viruses, and it turns out there are four essential conditions for the success of a virus. One of them is there's got to be a community of similar systems that run similar code. Second is there has to be sharing in that population. Third is there has to be storage in which the virus can replicate itself and to which it has access. And the last is it has to have some way to get itself executed. If you have all four of those, then a virus can succeed. One of the reasons that viruses don't succeed in the IOS thing is that Steve Jobs was willing to say we can constrain that, we're going to hide storage. And the execution business, the ultimate way you get yourself executed if all else fails, is you give bait to the user for him to click on. What we're seeing right now, if you read the Verizon Data Brief Report, right after default passwords, the next best way for the attacker to get control is with bait, fishing thing. That's turning out to the dominant and most successful attack right now. Ken Weiss gets on the list because he did secure ID, and secure ID was the first really successful strong authentication product. We can't live without strong authentication. We can't live without good key management. The best contribution, the real contribution of RSA is to key match, that's what we use it for. But the properties of RSA, the ability to attribute anything that was encrypted to the private key to the owner of that key, all of that stuff turns out to be just; RSA solves all kinds of really, really hard problems. And it's been influential in all open crypto systems. The most successful two crypto systems are SSL and Lotus Notes. Lotus Notes doesn't use RSA because it's; I think in fact it does use RSA; it does. It does use RSA for high level key management, so even Lotus Notes, a closed system, couldn't operate without RSA. We use RSA to share high level keys in Lotus Notes, so if I want my Lotus Notes domain to talk to your Lotus Notes domain, you and I meet over lunch and exchange public keys. After that, we put that into Lotus Notes, and; properties of good key management; fully automated, hidden from the user. Separation of keys. IBM people will tell you that one of the most significant contributions they made to key management — and they invented automatic key management — was what they call the control vector. This is a property of the key which says what it can be used for. So if it's used as a key encrypting key, it can't be used to encrypt data because that would weaken it. The control vector is what controls that. The control vector says this is what this key can be used for; this is the nature of this key; this is what kind of a key this is; and that's an invention that can't be attributed to an individual, I think it goes to the team.

Yost: Was James Anderson among . . .

Murray: Well, Jim Anderson, he's in the pantheon; he's in the pantheon along with Bob Abbott. The greatest compliment I ever got was I had a 20-minute slot to talk about crypto at some conference in New York City, and Jim Anderson was in the room; that was the second best compliment I ever got on crypto was Jim Anderson saying that in 20 minutes I didn't make any errors. I didn't commit any errors and I said some useful stuff. The other one was I did a customer presentation on crypto one time with Jonathan, and this was an hour or an hour and a half, so I covered a lot of ground. Jonathan Osias was on the telephone listening in; he was supposed to have done the presentation and couldn't be there. So he was on the phone listening in and I made the presentation, and he said I got it all right. So those two things are very, very important compliments. But to have Jim Anderson say you got it right was a really big compliment.

Yost: Did you regularly attend both the National Security Conference and the IEEE Symposium?

Murray: The IEEE Symposium was essentially closed, and I got invited to it a few times; I probably went three, four times. I went to every National Computer Security Conference and was probably on the program for most of them. I was sorry to see that program end. It ended when its champion, Jack Halloran retired from NSA.

Yost: And what year was that?

Murray: Oh, god. Well, it would've been in the mid-1990s because at the last National Computer Security Conference we formed the Colloquium for Information System Security Education and it just had its 17th meeting; so it would've been 1995, 1996, somewhere in there that it ended. I have some great memories from those conferences. One of my colleagues — Don Ingraham, he was an assistant DA in Oakland, California and he had a lot of the really early computer cases — he spoke at the National Computer Conference one time and he said, you IT people are never, ever going to be professionals until you stop paying criminals for after-dinner confessions.

Donn Parker and I had a pact between us that we would never appear on a program with a hacker. What was the guy's name? Bill Landreth. After we had agreed to appear at a conference in the Midwest, Donn and I both, this guy — his book was called *Out of the Inner Circle*, it was one of the early hacker books — and he was put on the program late. And we agreed than that we would not appear with him. It turned out that he did not show up and has never been seen or heard from since.

Another hero is Phil Zimmerman. Phil Zimmerman is a very courageous man. He bucked the NSA and has done; you know, the only crypto implementations that you can be pretty certain the NSA hasn't bugged are the things that Zimmerman's associated with. He had to put up with a lot of crap in order to do what he did. So he's not only a great; oh, and one of the things that he wrote, which turned out to be really significant was when he first published PGP — PGP stands for Pretty Good Privacy — and his point was an important point, which is that all crypto can really do for you is to get the security up to the level of your worst security; your next-worst security. So it can bring the middle up to the endpoints, okay? It can bring the security of stored data up to the security of where you store the key but no higher than that. So he wrote this really great paper that went with the first edition of PGP that said these are all the weaknesses in the use of cryptography; these are all the things that can get you in trouble. That turned out to me, to be almost as valuable as the code itself; was to be able to describe succinctly and in terms that anybody could understand, these are the limitations of your product. I like to say that crypto was the only security mechanism that we have that is orders of magnitude stronger than it needs to be. But its limitations are “broken fingers.” [Laughs.] If I want to know the key, and you know the pass phrase, I just break your fingers one after another until you tell me the pass phrase. Now, the limitation to that, of course, is I might not want people to know that I coerced you to get the key. I don’t want you to have evidence; when you walk out of the room I don’t want you bloody. I want both of us to be able to deny that you gave me the key or that I engaged in coercion. Fortunately for all of us, there is a limit to how much NSA is willing to coerce people but don’t test them.

One of my assumptions is NSA can read any message that it wants to. I tell every client that. If NSA is your adversary, trust me, if they want to know what your message says, they can find out what your message says. They can read every message that they want to.

One of the reasons that the NSA doesn’t like the public use of cryptography is that it reduces the efficiency of signals intelligence. It doesn’t prevent it, it just makes it more expensive than it would otherwise be. The network, most of the important traffic on the internet, is encrypted. It's encrypted under SSL. Can they recover message encrypted under SSL? Yeah. Can they identify all the messages that they might want to recover encrypted under SSL? Probably not. It just makes their whole job much, much tougher than it would ordinarily be so they don't really like it.

They got their dupe, Director Louis Freeh, of the FBI, to testify before Congress that strong crypto constituted "perfect security" for the criminal. He testified to that over, and over, and over again. Somebody must have been whispering in his ears that it wasn't true. Certainly the NSA knew; they did not so testify. The best strong crypto can do is raise it up to the level of your next-weakest security mechanism and that's all it can do. The FBI has demonstrated that over, and over, and over again. They corrupt systems, they corrupt people, and they find out eventually what they want to know.

But, with all of its limitations, it's the only thing that we've got that we can make arbitrarily strong. We can't make anything else arbitrarily strong. The limitations are in the implementation.

Brian Snow, after he retired, at an NSA Conference said NSA spends at least as much on implementations and systems, as on codes and ciphers. He was cagey about it; they are always cagey, NSA people. One need only watch their testimony before congressional committees to know that they have raised deception to a fine art.

Modern codes, modern ciphers are really, really, really strong. I've got a bunch of videos on the internet that I hope will stand the test of time, but one of them is about AES [Advanced Encryption Standard]. Big headline in the papers: AES broken. Well, you can't say that AES is broken because there is no standard. AES simply is. It is what it is, and it is what we know about it. Now, we know since day one is the most expensive attack against AES was an exhaustive attack against the key. This attack is guaranteed to work and no one would spend more than this. We had not known whether or not there was a cheaper attack. Turns out, some mathematicians demonstrated that you could recover a key from AES in one-fifth the time of a brute force attack. Is that broken? Until this work was done we didn't know that there was not a cheaper attack than brute force for AES, now we know that there is. On the other hand, if you had started that attack at the Big Bang, it still would not have completed. Is that broken? Not by any rational standard.

And one of Courtney's three laws; I have got to quote Courtney's three laws down here. Courtney's first law says nothing useful can be said about the security of a mechanism except in the context of a specific application and a specific environment; environment to include threat. His second law says never spend more money mitigating a risk than tolerating it will cost you. Those two keep me out of a lot of trouble; really important for me, otherwise I'd make all kinds of gross errors. The third one says there are management solutions for technical problems but there are no technical solutions for management problems. Courtney's three laws; absolutely brilliant. I may not owe the success of my career to Courtney, but I certainly owe the absence of failure to him. Those three laws have kept me out of a lot of trouble.

One more check to make sure I didn't leave somebody off. Oh, Ruth Davis. Ruth Davis goes on there because of courage. She had the vision to see what was needed, and the courage to pursue it, in a government that was just really gonna make it difficult for her. Multics, MVS, the M360, RSA secure ID. The great men, the seminal products — the Burroughs B5000 probably goes on that list for virtual memory. Okay. I didn't forget any of the [pause]

Yost: One person that I wanted to ask you about who was at IBM for a time period that we didn't get a chance to interview was Paul Karger. Did you interact with him much and can you comment on him?

Murray: Paul Karger was at Rancho Santa Fe. That was where we met. The only disagreement we ever had was over the concept of attack surface. No, no, there was another one he didn't like. Attack surface was one and there was one more. He did not think that the concept of attack surface was useful and I thought that it was. I thought that the more systems you had in the network the more vulnerable it was going to be, and the more places you could simply attack it looking for something. On the other hand, it also provided you with the ability to hide stuff. I can hide ports in use among 64,000 ports. It takes a long time to scan 64,000 ports and I'm never going to use more than 1,000 of them, okay? So I distribute the used ports in that 64,000 space and I can hide a lot of stuff in there. There is a concept that's called security by obscurity and it's used to disparage things. I say no, obscurity is the ultimate security mechanism. I'm always going to have at least one small secret that I must obscure; it may be a root key, it may be a pass phrase; it may be a PIN; but I'm always going to have some root secret that I have to obscure. And so the idea of security by obscurity is not a universal disparagement for me. All I do when I encrypt something is I reduce the problem of hiding that whole thing to the one of hiding the key. That's all encryption really does for me. Now, it facilitates exchange because I can ensure that I've had a secure key exchange before I secure the object. Before I transmit the object I want to make sure that only the receiver and I have the key, it has that second advantage. But ultimately, all it really does is reduce the size of the thing that I have to keep secret; that's the very best it can do for me. Before lunch is over, I will have remembered the other thing that we had a disagreement over. Mostly I think these were probably semantic things. But right before he died, there was something that he was having a problem with the whole community over; where he thought we all had it wrong.

Yost: Did his project with Lipner at DEC seeking to build an A1 have any significant influence on IBM in its strategic planning, with regard to computer security?

Murray: Probably not, but that was our mistake, not theirs. Olsen asked why anybody punch data into the card? Or have the keyboard attacks to the card, rather than attacks to the processor or the storage? And IBM's reaction to what they then called the minicomputer, they just missed it, right? They did some minis; they did lots of them — as a matter of fact, the universal channel control was a minicomputer — but they were all purpose built. For a narrow niche market they would do minicomputer, but the general purpose mini, they just didn't get it. And they missed it. And they missed the word processor, you know, Wang had to introduce the word processor.

Yost: So you don't place the AS/400 and its predecessors in the midrange series, in the mini category?

Murray: No, they are midrange, but they weren't directly competing with DEC. DEC was an order of magnitude scale smaller than us. They eventually started to scale up from their successes, but they were sold in a different marketplace. They were really selling into the general scientific marketplace whereas the AS/400 and the System 38 were being sold into the business application marketplace, for which they were ideally suited provided you didn't have data you wanted to import. All the data was going to be created and processed within that system. It's a great solution; and it's still the most secure operating system on the market. It's gorgeous.

You know, there are all kinds of wonderful properties of that system. It's single address space, it's symbolic-only addressing. The API can't get to a physical address, all it can do is use a symbolic address, and if the symbolic address is not known it'll either be defined or it's not going to happen. It was a finite state machine; an operation in the AS/400 will always take the machine from one correct state to another correct state. There was no such thing as an ABEND [abnormal termination] or a data exception [illegal operation on a data or encoding type]. Those things couldn't happen in an AS/400, they were just undefined operations. Everything that could happen was rigorously defined.

You could replace a program but you couldn't change it. Any change to a program automatically resulted in a change in its name and if the property of the object was data, no way in the world it could get executed, so it couldn't contaminate its procedures. Just a beautiful, beautiful architecture, but it relied on the fact that it was closed. In the marketplace, if I'm going to start something from scratch, closed system works. But if I want to replace existing applications, if I've got any legacy stuff that I want to [pause]; and that's what the market says.

Yost: Thank you so much, this is fascinating.

Murray: It's been fun. Nothing I enjoy as much as telling stories; reminiscing about old friends. [Laughs.]

Yost: It's extremely useful for our project and we greatly appreciate it.

Murray: Okay. I'm glad it's going to be in a permanent archive.