An Interview with

KARL LEVITT

OH 455

Conducted by Jeffrey R. Yost

on

6 June 2013

Computer Security History Project

Davis, California

Charles Babbage Institute
Center for the History of Information Technology
University of Minnesota, Minneapolis

Karl Levitt Interview

6 June 2013

Oral History 455

Abstract

Computer security pioneer discusses his educational background and early career, with the bulk of the interview on his approximately four decades plus focus on computer security research. He discusses his work with fault-tolerant systems, the institutional setting and his research and management roles at SRI (including PSOS, IDES/NIDES, etc.). He also discusses joining the faculty at UC-Davis in Computer Science and launching its Computer Security Laboratory, which has been influential in educating and placing computer security specialists in academe, government, and industry. Also discussed is his post-IDES/NIDES work on intrusion detection, voting systems and security, federal funding for computer security, his program manager role for Trustworthy Computing at NSF, and other topics.

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

Yost: My name is Jeffrey Yost from the University of Minnesota, the Charles Babbage Institute, and I'm here today on the campus of UC-Davis with Professor Karl Levitt. This is for CBI's NSF-funded project, "Building an Infrastructure for a Computer Security History." Karl, I'd like to begin with just some basic biographical questions. Could you tell me when and where you were born?

Levitt: I was born in the Bronx, New York, 1939. You said when and where? Okay.

Yost: And then you grew up there as well?

Levitt: Yes, I did. I went to the Bronx High School of Science, which people have heard of.

Yost: As a primary and secondary school student, what were your principle interests?

Levitt: Well, academic interests— maybe I knew from the time I was seven, I wanted to be an engineer.

Yost: That's pretty early.

Levitt: Well, partly because my parents — they grew up pre-Depression, but they survived the Depression — said it's a good job, it's steady, you make a good income, and there's tremendous need for them.

Yost: Knowing that you wanted to be an engineer at such a young age, did you think about what type of engineer you wanted to be?

Levitt: Well, my father's an electrician so that persuaded me to think about electrical engineering, but it's different than being an electrician for sure.

Yost: Definitely. And can you tell me about how you came to decide to do your undergrad at the Cooper Union?

Levitt: Well, in those days, and up until this year, it's a free private school. So, a private school wasn't an option, based on my parents' income. So it was either Cooper Union or the City College of New York. City College of New York is fine, but Cooper Union has a better rating, smaller school, and it was free.

Yost: I did not know that about Cooper Union, that it is free school, or was then.

Levitt: Yes, well, in fact just this year they sought to charge tuition for incoming freshmen after being free since 1859. So it's a big change and a lot of student protests about it.

Yost: I can imagine. So you took an electrical engineering curriculum at Cooper Union.

Levitt: That's correct.

Yost: Did you have any exposure to computers while there?

Levitt: Remarkably little. [Laughs.] We got exposed to an analog computer while we were there. So I got the chance to plug operational amplifiers, and try out various simple problems. We didn't have a digital computer at Cooper Union in those days.

Yost: What did you think about that experience, in using that analog computer?

Levitt: I don't think. Wow.

Yost: Did it have an impression on you?

Levitt: No, very little.

Yost: Okay.

Levitt: I wasn't impressionable enough to think ahead and think what this might all mean some day. [Laughs.] It just seemed to me like any kind of electronic device, like an amplifier or something like that. Ordinary electronic gear; nothing impressive at all.

Yost: Did you go straight on to grad school at U Conn?

Levitt: No. Well, I went to U Conn at night but I was working at United Aircraft during the day because I had just gotten married at the ripe old age of 20. And oh, yes, we had a child two years later, so I had a family to support. So I went to U Conn at night and I worked on electronic circuit design at United Aircraft.

Yost: Did you use computers at all at United Aircraft?

Levitt: No. We sort of designed special purpose gear that one can think of as computers but the display and the like — yes, just basically designed very large circuits for various things like I designed an infrared camera, and a display controller. They used computer principles but they weren't general purpose computers, though, by any means.

Yost: From the date on the records I gathered, you graduated from Cooper Union in 1960, and that's the same year that you started at United Aircraft.

Levitt: That's right. And the same year that I started at the University of Connecticut.

Yost: You completed your master's in engineering in 1962.

Levitt: Correct.

Yost: Did you do a master's thesis for your master's degree?

Levitt: No, it was a satellite program, so we were just required to take classes. I took a few classes in computer design, assembly language programming, and so that was my first introduction to computers.

Yost: Was there a computing center at U Conn at that time?

Levitt: Well, yes. It was sort of inside glass doors and it was; what was it? It was probably they had an [IBM] 1620 and maybe a 1790, I guess.

Yost: So it was the operators that were behind glass doors, they were the ones that directly interacted with the machine in a batch environment.

Levitt: Oh, yes. You had a deck of cards, basically. Exactly.

Yost: And you mentioned you took a course in assembly language. Is that your only computer—

Levitt: Well, I took classes in digital circuit design, and learnt about minimization and Quine-McCluskey. I don't know if you remember that, but — well, you're too young to remember that. But it was all the rage back then. Relay circuits and transistors, minimize the number of components. It was kind of fun, you know? I didn't appreciate that these were NP-hard problems

back then. I didn't know what that meant. Teachers knew what it meant but I got the feeling though that they weren't easy.

Yost: And what were you thinking about career-wise back then?

Levitt: Well, I didn't think I could do circuit design the rest of my career and make a living doing it. Things were changing. It wasn't VLSI — my goodness — it was integrated circuits were just coming, back then. That seemed like fun, but still, a sense that things were going to change. One of my teachers called it, saying — as a person who had a pocket with a couple of capacitors, a pocket with transistors, a pocket with inductors, and a pocket of resistors — sensing what was going to be a dinosaur some day. Okay, that could be me.

Yost: So in wanting to a pursue a Ph.D. were you thinking about an academic career, at that point?

Levitt: Yes and no. Thinking of a research career. Academics certainly seemed on the horizon, then. Yes, it seemed like fun, okay?

Yost: Tell me about your decision to continue your education for a Ph.D. at NYU.

Levitt: Let's see, it was kind of just almost a lucky choice. Some people said why are you going there? They haven't graduated any Ph.D.s in electronics there yet. [Laughs.] But it was a young faculty and they were very energetic. And in some new fields coming up, the theory of circuit

design was just coming up and information theory was just starting. It wasn't that many years after Shannon. But error correcting codes was something that really seemed exciting. So I did that and I just lucked into getting an advisor who was an expert in that.

Yost: Were you involved with the Courant Institute?

Levitt: I didn't even know about the Courant Institute. NYU then had an uptown campus with its engineering and the downtown campus which included the Courant Institute. I took a class at the Courant and that was very exciting, but when I started I didn't even know it existed.

Yost: Can you talk about the courses that you took; the most influential courses?

Levitt: Certainly information theory. Information theory, coding theory, communication theory, and a bit more on circuit design, digital circuit design. That was beginning to be more of an academic discipline then so those all influenced me. But my advisor was great. He just passed away. His name was Jack Wolf. In fact, Peter Neumann knows him. My guess is that it may come up in the conversation but he's very good. He's very influential in the field of error correcting codes. And he worked with Dave Huffman, who was certainly famous in this area and passed away recently, and David Slepian, and Wyner.

Yost: Huffman, of course, was influential to Peter.

Levitt: Oh really. Okay. Ah.

Yost: He talked quite a lot about Huffman.

Levitt: Okay. And Jake and Ziv, who's essentially the father of the Ziv-algorithm. Okay, yes. So most of those people were at Bell Labs, in those days. Peter probably talked about his time at Bell Labs. I'm sure he did. Bell Labs was extremely influential in those days.

Yost: Yes. And at Courant, were there any faculty members that were influential to you?

Levitt: Yes, my goodness. His name is Wilhelm Magnus and I learned some sort of abstract algebra from him. He's just a delightful, delightful guy. I learned group theory, rings, and fields. Yes, that was very influential in my career.

Yost: Your dissertation was on entitled, "Correlation Properties of Multilevel Cyclic Sequences."

Levitt: Right.

Yost: Can you describe both how you came to that research topic, as well as give a brief summary of the dissertation?

Levitt: Sure. So my advisor had worked in error correcting codes, and cyclic sequences had just come about then. There's something called maximal length cyclic sequences, and these were

sequences that you'd use for communication. And a very interesting property that if you looked at the correlation properties under it, there's an order correlation. So if you took the sequence and you shifted it by any number of bits up to the length of it, they'd be uncorrelated. So therefore, you could use this — it was very good for secret communications because if you shifted it a bit, you had to be perfectly synchronized in order to get the information content out of it. So I looked at sets of these sequences so that you can use them, especially for communication. And my advisor helped me an awful lot with this. It was fun, and I got to use computers to do simulations, to determine how well they'd behave in the presence of various noise. So I used what I could learn to be better with correcting code, communication theory, which is most of the classes that I took at NYU.

Yost: What computer systems did you use for that?

Levitt: It was maybe an IBM 790 or 794. So I guess I had — it wasn't a PDP-8, it was PDP-5 — the predecessor of PDP-8 all to myself to do these simulations. That was fun. Those were the days of paper tape so you'd actually type in your source program into a paper tape and there wasn't much memory so you'd run a compiler tape through it. Okay, there wasn't enough stored memory so you'd have to store the tape or run a tape through that has the compiler on it. And then run your source program through [and] get another tape, which is your object program, and then put in the operating system — I guess it was FORTRAN — and then run your program and get the results on tape, and take it over to the printer. [Laughs.] It was very painful if you made mistakes. You learned how to make very few mistakes, okay? You learned to be very careful before you ran a program. Very different from now, of course.

Yost: You mentioned your primary advisor. Were there others on your committee who were particularly influential to you?

Levitt: Yes. Ludwik Kurz. I've kind of lost track of him, but he was focused on communication theory. And then the person who was dean here, Mohammed Kousi, he wasn't on my thesis committee he was on my exam committee. He was tough, but very good in circuit theory; an expert in circuit theory. I actually bumped into him a couple times over the years. And then, I guess we're a little ahead of the story, but he was dean here in the mid-1980s and he persuaded me to come here.

Yost: As you finished your dissertation, can you talk about your job search and how you ended up at SRI?

Levitt: Sure. It was a time when there were jobs galore for Ph.D.s. I like to say I certainly wasn't a war baby, I was a Depression-age baby and there weren't enough of us then and so there were loads of jobs. So I had offers from Purdue, and Illinois, but my wife said she didn't want to live in the middle of the country. I had offers from Bell Labs and then JPL, and then SRI. And we decided gee, this is probably — well, I'm going to say professionally and personally this was attractive. Personal, we'd barely been west of the Hudson and felt maybe it was time to try a different part of the country. And there were some things happening and SRI, it was fault-tolerant computing which was just beginning to start up. NASA was very interested in that topic because obviously, they wanted reliability. So that sounded like something I really wanted to do.

And there were people who did work on error correcting codes. Bernie Elspas — I don't know if Peter Neumann mentioned him but he's an old time coding theory guy. He still lives in Palo Alto; I'm still friends with him. So he said come, we can work on coding theory. And then switching theory, I guess it was going to die soon as a subject. Again, it dealt with design of circuits, special purpose circuits, computers, and there was minimization, but also other constraints came into play. If you were thinking of putting them on a chip well, you might have to minimize the number of connections. So a very interesting constraint problems. They don't show up so much now, but they were fascinating problems in those days. So all this was interesting to me, as well as the personal attraction of Northern California which was pre-Silicon Valley days. So it was mostly what you'd expect for Palo Alto and Menlo Park. And San Francisco, of course.

Yost: Of course, a wonderful climate and environment.

Levitt: Still a wonderful climate in many ways, yes.

Yost: So in addition to Bernie and Peter, there was Jack Goldberg?

Levitt: Jack Goldberg, who's an old-time fault-tolerance person. Yes, I'm still friends with him. He lives in Palo Alto. Someone named Bill Kautz who's sort of been lost in the annals of history, but who was very good at switching theory and is a very, very smart guy. Someone named Marshall Pease, who you may know about because he's written about discontinuous agreement. That'll come up in our conversation, but he did that many years later. So those were my

primaries. Those are all very good people and they couldn't really get young people. I guess that I was 25, in those days or 26. [Laughs.] There just weren't enough of us around.

Yost: Organizationally, how was SRI set up in those days?

Levitt: Almost the same as it is now.

Yost: There was a computer science laboratory?

Levitt: There was a computer science laboratory. It's one of the — Peter might have told you about this — maybe the longest-standing computer science laboratory in the country. I can't imagine anything that would be longer standing than that.

Yost: Can you recall what year it was formed?

Levitt: I would guess in the 1950s. And they did Project ERMA. Did Peter talk to you about that?

Yost: Yes.

Levitt: You know about ERMA.

Yost: Oh, of course—knew that was SRI. I didn't realize it was out of the computer science lab.

Levitt: Yes, it came out of the computer science lab. Those people, you know, Jack Goldberg and Bernie Elspas, and Doug Engelbart. I'm sure you know about his involvement. It might've been his first assignment at SRI.

Yost: Dating back to the mid-1950s.

Levitt: Exactly. Right. A remarkable set of people. And it's also said the same way, I mean, you've gotta get money. There's no endowment at SRI. You've gotta go out and get funding for the work you want to do.

Yost: What was the first project that you worked on there?

Levitt: It dealt with fault-tolerant computing and principles of fault- tolerant computing, and the ways of doing majority voting, which goes back a long way. And we wanted to figure out okay, where do you place the voters to maximize reliability and minimize the amount of extra equipment, because you couldn't waste equipment. Hardware wasn't free, in terms of space or in terms of the cost, in those days. So that was my first assignment and that was enjoyable.

Yost: Speaking of equipment, can you talk about the systems SRI had at that time?

Levitt: The systems? [Long whistle.] Well, it was mostly IBM and DEC. So there were probably PDP-8s around and IBM mainframes, I guess. Time-sharing hadn't come about yet. And the

PDP-10 was just coming around at that time and that gave rise to time-sharing and completely changed the way we did things.

Yost: Did they have a IBM System 360?

Levitt: Probably, yes. When I joined, no. I joined in the mid-1960s and I don't think they had a 360 then.

Yost: In 1967, you taught at Stanford.

Levitt: Oh, yes that's right.

Yost: Can you talk about that experience and the people you interacted with?

Levitt: I taught a class in error-correcting codes in part because they wanted a class in that and in part because I learned about it in my Ph.D. days and had been working in this area. Who did I interact with? Well, Tom Cover who — I think he recently passed away — did work in more theoretical information theory. And then Ed McCluskey who I guess was one of the founders of fault-tolerant computing and certainly did work on digital computer design. Those are the two main people I interacted with then.

Yost: That first fault-tolerant project, was that an existing contract that you joined?

Levitt: Ishak and I joined, yes. It had to be with NASA. It was, it was with the NASA electronics research center, which doesn't exist now. Ted Kennedy, I guess, helped fund that when he became senator in 1962. I don't think it existed more than 10 years or so with various cost cutting measures. It just got absorbed into other centers. But there were good people in it and yes, perhaps NASA should have a electronics center, you think? In those days, it was very important to NASA.

Yost: Was there freedom at SRI in the early years to do some exploratory research before getting a contract?

Levitt: Yes, yes there was. Probably more so then than now. But then maybe that's not fair. I don't know enough about SRI now. Yes, definitely. So typically you would charge four days a week to the contract and have a day a week plus whatever of your own time you wanted to put into it to do exploratories. The idea was to help get contracts, of course.

Yost: Develop expertise and contacts for the next contract?

Levitt: Exactly. Those were the days before SRI became sort of entrepreneurial, I guess. We'll talk about that.

Yost: How many people were in the computer science lab in the mid-1960s, can you tell me?

Levitt: Maybe a dozen. I should also mention, if you want me to, that I had a chance to work with Doug Engelbart then. And Bill English, do you know that name?

Yost: Yes.

Levitt: I knew you would. Yes, so that was when Doug was just starting up his vision. So time-sharing had just come around, so he had, my guess is a PDP-10 and was using it in a time-sharing system but he wanted to make it faster. At that time it took a lot of computer time just to drive displays that you had, so he had the vision — it wasn't my idea — that we put this into hardware, the display. The hardware could go much faster rather than just making programs. So he asked me to help him design a display controller, which I did. So I had a chance to work with Doug and experience the environment at that organization. I'll tell you some things off the record, if you like, okay? [Laughs.] But anyway, he's fantastic. We used to jog together at lunch time. I guess the mouse already existed and the five-fingered keyboard. Have you seen that?

Yost: Yes.

Levitt: Okay. So those already existed at the time.

Yost: Were you present for his famous presentation?

Levitt: I wasn't present but I was around as he was preparing for it. But a very calm guy.

Yost: Can you talk a bit about working with Bill English?

Levitt: I primarily tracked with Bill, even though I jogged with Doug. Bill was more of a designer, okay? And so I don't think he gets enough credit for being creative. He was sort of the bridge between Doug and these young kids. Doug wasn't old then, of course not. But he was the person that kept the kids in line and I learned a lot from him. I didn't know enough about hardware design. I knew the theory, but I didn't know it from doing it in practice. As I said, he made things work, okay? That's what I'd say. Doug was the vision. When we couldn't get things to work, we had Bill. It was basically Doug, Bill, and the kids, was what the organization was.

Yost: And who was director of the lab?

Levitt: I was involved with two labs. I worked with Jack Goldberg in my lab, the computer science lab, which still exists. Doug was head of the Augmentation Research Center, ARC, which does not exist anymore. Bill was, I guess, his assistant or his associate. I mean, Jack Goldberg was my primary boss but I helped Doug out on this.

Yost: In those early years at SRI, in the 1960s, were you attending professional conferences?

Levitt: Oh yes.

Yost: Which ones?

Levitt: I'm dating myself. [Laughs.] There weren't many conferences in those days that they had. There were the Joint Computer Conferences. Spring Joint and Fall Joint Computer Conferences. So you get all collection of papers ranging from theory to practice. It was 10,000 people. I attended one in Atlantic City and I thought there must be 10,000 people there. San Francisco — wasn't Moscone Center, it was the Civic Center of San Francisco. And eventually got to people from Boston, which is incredible, if you think about it, right? So they held them in DC, or Baltimore, Atlantic City. Fall Joint was West coast; Spring Joint was East coast. And information theory had the Salida conference, started then or maybe it was a couple years old. But there weren't many. I'd say maybe half a dozen annual conferences in those days and a couple journals. It was very easy. People say, kidding me, they used to say, God, you know everything in computer science. It wasn't hard, actually. Of course now it's impossible.

Yost: At the Spring Joint Computer Conference in 1967 — I believe it was Spring rather than the Fall — Willis Ware, from RAND of course, and Bernard Peters, from NSA, gave a talk on multilevel security. I was wondering, did you hear that talk and were you thinking at all about multilevel security at that time?

Levitt: I heard it; it didn't resonate at all to me. [Laughs.] It didn't. I wasn't thinking security then. We'll get to it later. I was thinking fault tolerance, which sort of gave rise to security; thinking error correcting codes, which has some bearing on cryptography although the math is different; and Shannon information theory, of course, applies above. But I wasn't thinking multilevel security then.

Yost: What about at the end of the decade? Ware's Defense Science Board, the famous Ware Report comes out in 1970.

Levitt: Yes, I was aware of it but it didn't have an impact on me yet. I still wasn't thinking security.

Yost: Were any of your colleagues at SRI, that you know of, influenced by any of this? Really, it's more kind of definition of a problem at hand with multilevel security than solutions, at that time.

Levitt: I would say no. Maybe I'm not being fair to them. I can ask them about it when I see them next, but I don't think they were. Peter had not joined us yet at SRI.

Yost: Right. Peter, of course being part of Multics group while at Bell Labs, was [pause]

Levitt: Right, that's it exactly. He was thinking about separation and access control. I was not thinking that at all then.

Yost: In 1968-69, you were simultaneously at SRI and T.J. Watson lab?

Levitt: I left SRI in 1968 in part for personal reasons. My wife decided she missed... Calls to the East coast were very expensive then, business calls and thought, okay, so we want to go back. So I went to T.J. Watson for about nine months. I didn't enjoy it that much. I could've stuck it out

but I sort of missed SRI. SRI just has some excitement to it, okay? I can ask you a question. Did you hear that program on NPR, John Markoff had about SRI?

Yost: No.

Levitt: I'll send you a link to it. It wasn't that good but anyway, it's changed in some ways.

Yost: He's been the journalist that has written more on security, there's Levy's book also, but Markoff has written more on the topic over time.

Levitt: Yes. It emphasized more the entrepreneurial aspect of SRI; how it tried to influence companies and make money and I think that wasn't the issue when I was there, at all. And I think the most people, that doesn't drive them. What drives them is the science. So anyway, I don't think that program does justice to SRI. That's a personal opinion, okay? I can send you a link to it.

Yost: Great. I'll look for it. Can you talk a little bit more about the culture of Watson Lab and how it differed from SRI?

Levitt: Yes. Maybe I was just too immature to appreciate it. IBM wasn't struggling then, I mean, they were the leader in the field. They were a little worried that their products didn't reflect the technology that's coming out of Watson. So I worked on demand paging. IBM came with 360, and that was meant to be a system that would work from, I don't know, maybe a micro center

[or] at least, minis up to mainframes. A very general purpose operating system in which you could segment it to solve various problems. It did fairly well at the low levels, but it failed miserably for time-sharing, in part because — well, a number of things. One is there just wasn't enough memory. Memory was too expensive in those days. And second, because people didn't understand operating systems in those days. Think about it. This is before the notion of the process, and scheduling wasn't really known that much, what to do about it. And so I worked on sort of a demand paging, trying to make paging more efficient. Tremendously hard problem and I didn't appreciate all the constraints on it. But eventually memory became cheaper and Moore's Law came into play. And things worked, and people know how to design operating systems now. And in fact, it was interesting — I don't know if Peter talked to you about it — but software engineering was just beginning to come into play then. In 1968 there was a conference in Germany. I've forgotten where it was.

Yost: Garmisch?

Levitt: Garmisch, that's right. Exactly. I remember I didn't attend it but I carefully read the proceedings of it, and they said the operating system is the most complicated thing man will ever design. More complicated than submarines. [Laughs.] And they were, because people didn't know how to structure them. And now I teach operating systems in a quarter. I teach an operating system in a quarter because now we have principles, we actually design structures. But anyway, the culture of IBM — it was some very smart people there, yet too much freedom, actually. Because IBM was just loaded with money, you could work on whatever you wanted, and if you helped out IBM all the better. Occasionally, they'd come in and say hey, can you

work on this problem? So they sent me to various places for day trips to try to help out on problems with their Federal Systems Division, with fault-tolerant computing. That was good. I got to learn. But I felt that this wasn't enough group activity. I think it was my immaturity. SRI seemed more exciting because you have to go and get money, and that was sort of a forcing function. It forced you to think about the future, think about what problems are important, and to form relationships with your colleagues. And then go out and sell your work. And then be told no, that's not my problem, by the sponsor.

Yost: At IBM research, there was, at the time, tremendous freedom to explore.

Levitt: Yes, which was great.

Yost: I know that changed drastically over time.

Levitt: Yes it has.

Yost: And it also did at Bell Labs.

Levitt: Right. Exactly.

Yost: They lost some talent at IBM as it became less focused on basic research, I think when Edgar Codd left IBM research, he cited that as a reason.

Levitt: That's right, exactly.

Yost: . . . changing culture towards a focus on products . . .

Levitt: I didn't track him; he's in California. I didn't track him, but he was just working on relational databases at the time so there were exciting things going on there at the time.

Yost: And in San Jose, of course, RAMAC was an exciting development.

Levitt: Right, exactly.

Yost: You talked about fault-tolerant work. Were there other projects that predate getting involved in the Provably Secure Operating System [PSOS] at SRI that you worked on?

Levitt: Yes. So when I get back to SRI, we get a call from somebody at the CIA and he had just heard about Bob Floyd's paper on — it involved proving the correctness of programs. I can't remember the exact title; you probably know the paper, an extremely influential paper in the 1960s. And John McCarthy wrote some papers on proving correctness, proving properties of LISP programs. And Tony Hoare had just written a paper coming up with what we call the Hoare Axiom on Semantics. Those are three classic papers that came out in the 1960s. I got a call from somebody at the CIA saying I just read these papers on proving correctness of programs; we have programs that our analysts are doing that seem to have bugs in them, and they're causing

analysts to make mistakes. I knew nothing about the topic but I said okay. So I wrote a proposal then. I guess I was the leader. You might know Cordell Green? You know that name?

Yost: No, I don't.

Levitt: Oh, he's the head of Kestrel Institute in Palo Alto. At that time I think he had just gotten his Ph.D. at Stanford and he was about to go off to DARPA because he was ROTC, so he had to serve. But he and I wrote the proposal. And then Richard Waldinger, do you know that name?

Yost: I've heard of him.

Levitt: Okay, he was at SRI; he's still there. This was 1969, and he had just gotten a Ph.D. at CMU [Carnegie Mellon University]. It night have been Newell, or Herb Simon. Together, we wrote a proposal. I didn't know what I was doing or getting myself into, but yes, [we] wrote the proposal and we got the funding. They funded us for three years. It gave rise to a very interesting system and Rich Waldinger's 90 percent responsible for it. It was called Reasoning Math Programs that Rich and I wrote, but he was mostly responsible for it. And Bernie Elspas helped us. It was based on a theorem prover that Cordell Green actually came up with. It's called QA4, Question Answering 4. Question answering was all the rage in those days. You wanted to be able to pose questions to the program, you get answers back based upon its knowledge base and rules of inferences. It was a very flexible system. You wouldn't build a theorem prover like that today because it was just very ad hoc. We didn't quite know what we were doing but were able to prove some programs up that were maybe a hundred lines. The sponsor got frustrated. He didn't

say it's getting to the thousand-line programs very soon, let alone the million-line programs, but it gave rise to interesting work and ideas. Then Bernie Elspas and I wrote a survey paper on program verification. And again, I didn't know what I was doing, but it sounded fun. In those days, ACM actually paid you to write/submit papers because it had just started and they wanted to get decent papers. That doesn't happen now; you pay them. [Laughs.] But anyway, so that got me into program verification. That was fun and, of course, the field has taken off well beyond that. There are very smart people working in program verification now but that did give rise to computer PSOS.

Yost: How was that paper received by the community?

Levitt: I think it was well received. People looked at it as being ad hoc, and they were correct. But I say it checked the boxes off. We were able to take source programs and essentially compile them and generate what are called verification conditions, which are properties you have to prove for a program to be correct. And it showed you couldn't do those things automatically, that you needed sort of human guidance. In those days, human guidance was typically you added rules. The idea was you edit these rules and somehow a human could look at these rules, say the rules are correct, and you could trust the rules more than you could trust the program you were trying to prove because it was simpler and based upon mathematics. I think the tables were over safe. The field's gone way beyond that now.

Yost: What were the years of that project, do you recall?

Levitt: Sure. It was probably 1970 or 1973, I'm guessing. Something like that.

Yost: Will that take you right up to the start of PSOS?

Levitt: Yes, PSOS. Exactly.

Yost: Can you tell me about the origin of that project?

Levitt: Sure. I didn't prepare for this but I remember it very well. So, did Peter talk to you about a fellow named Ted Linden? Did that name come up?

Yost: Yes, I believe he mentioned him.

Levitt: Good. So he was at NSA then and we became very good friends. He passed away maybe 10 years ago. A very smart guy at NSA. Computer security, I guess, was just becoming of interest to NSA. This is pre-network, of course, so it was just strictly mainframes, time-sharing systems so they didn't have much activity in computer security. There was maybe half dozen people. Dan Edwards, do you know that name?

Yost: Yes, I'll be interviewing him in a couple of days.

Levitt: Oh, fantastic. He's a great guy.

Yost: I also would like to interview Hilda Faust.

Levitt: Wow. Say hi to both of them for me. So Hilda was head of the group. Ann Marmor-Squires came into that group a bit later; and Martha Branstad. I think Ted was the brains behind it, the force behind it. I said Dan Edwards was maybe thinking about security. He had some security principles then. I forgot when, it might have been just around the time that Saltzer and Schroeder came out with their principles. Daniel was very influential.

Yost: It was published in 1973 I think.

Levitt: Oh, so it was just coming out then. I think PSOS started in 1973. Anyway, Dan Pittman came to us, he said he wanted us to do it, and he wanted it to be verified. That gave me some connection with it because I knew some program verification. I knew fault tolerance, and in all honesty, I didn't know security at all. I didn't really know what access control was at the time. Information flow was something I was just learning about. But I'd learned a lot from Peter. So we got the project and it survived for a good number of years. Peter probably told you this, but a couple of people were very influential in the project. Bob Fabry, did he come up? Okay. Basically, he was the one who looked at capability-based addressing, and that came about following the work of Dennis and Van Horn. Bob Fabry knew that work. And then a fellow, Larry Robinson, did Peter mention his name to you?

Yost: I don't recall.

Levitt: Maybe he did. You have the recording. I think he would've anyway. So he was an odd duck. He'd just finished some graduate work at Carnegie Mellon working with Dave Parnas. I'm sure you know Dave Parnas. What was interesting then about our group at SRI was we had the old timers who did work on switching theory, and we really didn't have anybody who knew software and also system design because it was just coming into play then. I guess Peter knew it from his Multics days, but this kid, Larry Robinson came in, and he knew a lot. And he knew about principles and knew about information hiding from Parnas. So he came up with the idea and I sort of followed it along. The idea is that a multiple, not multilevel security, but layers of abstraction. So that was just coming into play there, Dave Parnas' idea. Others, well, Dijkstra, of course, the THE system, and then Larry Robinson. So we had capability-based addressing from Bob Fabry. We had the idea of layers of abstraction, and we said okay, why don't we design a system where we have multiple layers of abstraction with capabilities at each level. Then we became aware of the confinement problem, Lampson's confinement problem, so we decided to work on this multi-layer system. The idea then was multi layers are easier to verify. Yes, you could lay it out a layer at a time. Data abstraction was just coming into play then, too. Jim Horning, who recently passed away, was one of the people who did work on that; Tony Hoare did. The idea we came up with was a sort of a "methodology," HD, a hierarchical development methodology. And so we designed a multi-layer system and we did some proofs about it, but my friend Bob Boyer — you know that name, Robert Boyer? — okay, he was with us then. He'd gotten a degree I guess at Texas and worked with J. Moore, who is currently chair, or was chair at the University of Texas, the last I knew. Well anyway, Bob Boyer joined SRI, and we worked on verification of system. Bob Boyer used to call it "hopefully" secure operating system rather than provable secure, but we'll get into proof at some point later on. Anyway, we did some

proofs and I think the system was influential in some ways, in many ways. But basically, Schroeder would use capabilities, throughout that could give you access to any kind of object in the system whether it's memory, software defined objects, or whatever. At each level we did some proofs. So we proved that essentially capabilities were necessary and sufficient for getting access to an object. So it was, yes, it was fun. We stayed with it for three or four years at SRI. Maybe longer, maybe four or five years at SRI.

Yost: Can you talk about the degree at which Multics, in part through Peter, but also other people's study of and publications on Multics influenced you?

Levitt: Sure. Because, well certainly access control. Multics didn't have multilevel security but it dealt with discretionary access. And I guess it had some notion of capabilities in it, but definitely the layers and the rings of control. Certainly that influenced us. What else? Oh, segmentation, of course. Because then we're designing PSOS, we thought memory segmentation would be a certain level. So ours would be segments, and then below that we would have paging, because segments have to be paged. So certainly the idea of segments influenced us a great deal.

Yost: And simultaneous to this, about the start of the project, the Anderson Report comes out. Can you talk about your response to that report and potential influences it had on PSOS?

Levitt: Sure. Well, let me think for a minute. It's kind of confusing; there are various Anderson Reports.

Yost: Yes, there's a couple that came out, out the committee. And he, of course, published other papers after that.

Levitt: Certainly a later one on intrusion detection influenced us. Oh, reference monitors, I'm sorry.

Yost: Kernels and reference monitors—those were within the two-part Anderson Report.

Levitt: Kernels. Okay. We thought about it in several ways. We were trying to think that maybe we should design a kernel and then the idea was essentially all interaction would be through the kernel, and then only the kernel would have to be correct for your system to be correct. That didn't quite play with us because we're thinking about that there could be errors in different systems.

[INTERRUPTION]

Yes, this idea of a kernel. So we sort of rejected the idea of a kernel because we figured there's certain correctness probably you want at every level in the system. The kernel was fine for guaranteeing that objects could only access other objects, enforcing access control. Well it wasn't enough to establish that the process did the right thing. So in some sense, the lowest levels of PSOS we sided with kernel, but still the upper layers, which did more application stuff, would basically select correctness properties. But it was certainly, influential. Reference monitors mediating all references was certainly part of PSOS. Because for a capability we said you have the capability to an object, then essentially that was necessary and sufficient to give

you access to the object and the fact that you couldn't bypass the capability access mechanisms was influenced by the Anderson Report.

Yost: As you're working on the PSOS project, were there any security meetings that you were attending? Was the community beginning to interact in terms of meetings or just reading each other publications and reports?

Levitt: No, there were meetings. So I don't think there was any security; I guess the Oakland conference — when was the Oakland conference?

Yost: 1980 was the start, the first with available published proceedings I think was 1981.

Levitt: Thank you. I'm sure they just celebrated their 25th, or 30th anniversary, maybe? Something like that. Of course I can do the arithmetic, anyway. [Laughs.] So, no. Most of it's done; a very influential meeting was 1974 or 1975, the conference on dependable computing, I think it was. It was in Pasadena, and it was a very influential meeting. We presented PSOS there and there was also papers on verification testing; a very eclectic meeting. God, I almost forgot about that meeting. Yes, I think it was 1975. IFIP meetings still exist on this campus, actually.

Levitt: That's right. This was a very influential meeting. In fact, I think it gave rise to meeting on fault-tolerant computing, it sort of split off. I can't remember the name of this meeting; it's 1975, but we can look it up together at some point. We presented PSOS at that meeting and Dijkstra was there. He listened; he stayed the entire meeting. And then a year after that [pause]

Yost: Can you talk about the reception to the paper at that meeting?

Levitt: Wow. It's a good question. I don't think anybody went away saying this is the cat's meow, okay? Nobody said that. Good question. I'm trying to remember when it was. It might be 1974. I honestly can't remember. No, PSOS was 1975. That's true.

Yost: Do you recall any of the other papers at that meeting and how they may have influenced you?

Levitt: My goodness. Well, the papers on verification were coming out at the time, too. That sort of made the process more structured. Somebody named Jim King, from IBM, who I've lost track of over the years. He was at IBM and he presented a nice paper on verification, also testing. So yes, it sort of started. You probably know the DeMillo Lipton and Perlis papers? Okay. So I don't think they presented something there, but they were at that meeting. The guy at U-Mass, Leon Osterweil, do you know him? Okay. Anyway, he used to remember a fight in those days about verification versus testing? I said I don't remember us fighting, but anyway. [Laughs.] So it started more serious work on testing, and people saying why is it that we spend so much research effort on verification, which we rarely do, and so little research effort on testing, which we do all the time? So I think it gave rise to testing as a formal study. In fact, we had a paper there at the same meeting on testing. It was some things I did with Bob Boyer and Bernie Elspas, using symbolic evaluation to do testing, sort of work your way through the paths of the program with symbolic data. It was nothing exciting. Dijkstra heard that paper and then he grumbled the

whole time; I was just watching him grumbling. Later I heard him say it's the stupidest idea I ever heard. [Laughs.] Anyway. But I think he liked PSOS, because later on Larry Robinson and I visited him for two days in Eindhoven and talked to him about PSOS. If you want, I can talk about that.

Yost: Yes, definitely, please do.

Levitt: Okay. I don't remember all the details and I didn't appreciate it at the time but he said to us — I think he was intrigued by it but it's hard to say with Dijkstra. But anyway, he said why don't you just do this in a programming language? Make your programming language safe. Make your programming language enforce the access and that would make much more sense to me. Then you rely on your compiler instead of the operating system but you still need a run time system to make it work. He said he didn't buy it. We were actually inspired by the THE system, his multi-layer system. We certainly talked about that. But in the end, I don't think he liked the idea, necessarily. But I still think one needs a secure operating system. Okay. You need a run time system, in any event underneath, and that could be your operating system. But anyway, we spent two days with him. Larry Robinson and I spent two days with him and it was a grueling two days. I should say that he smoked like a smokestack.

Yost: What did you see as the greatest challenges to PSOS?

Levitt: Wow. Alright, so he goes back to Butler Lampson's idea of least privilege. He still talks about this. I don't think he mentioned PSOS very much, okay? There's no reason why he

should. It was managing the privileges. In PSOS the idea is you would create objects, and so the creator would have all rights to the objects and the creator would disperse objects to the various subjects and the question is how do you manage that? Essentially, who gets access to what? A particular subject, and it could be a program, or a user, or a network gets access only to what it needs and no more than that. Butler Lampson says — in fact, I had a huge talk with him, he's a fine guy, you should talk to him — and he said we've been there and it doesn't work. He says the management is just impossible. And I think he's right. It's very hard to control that and decide exactly what; I thought we had it structured enough but I'm not sure we did. So that's one. Number two is I think we talked about the proof and we never did it, we never fully did it. And I think it's still a major challenge. I think we can get to that later but there's still that proving operating systems is something that's — well, now people are trying to prove hypervisors, so even small pieces. So I think that's still a challenge. Number three was we started to think about safe programming languages then, but we never really did it so we came up with sort of experimental languages that were sort of type safe. This is pre-Java, of course. And so that was the challenge, and of course, C took off so we never [pause]. What else? I would guess the hardware. People still talk about tech architectures and we weren't the first to have a need [for] it, but that would really help because the idea was then that essentially, the hardware would take care of things. Well, there can be mistakes in hardware; there can be Trojan horses in hardware and the like and we can get into that when we talk about challenges of security. But we never, of course, succeed in that. We think about the Burroughs 5500, something that had tagging, so we were thinking about building on that. So I think those would be the main challenges: the management, the proofs, the hardware, and the language.

Yost: Shortly after you started the project, the papers from Bell and LaPadula had come out. Did those have an impact?

Levitt: Oh yes. Sure. Those were remarkable ideas. So initially when we designed it, we didn't have multilevel security in mind at all; we didn't know about multilevel security. Then it came up and, of course, Hilda and Ted Linden knew all about this, and Dan Edwards. So what came out then was that we said okay, we want to use capabilities. Larry Robinson said capabilities can do anything. So we said okay, that's our hammer. So we're going to solve multilevel security with capabilities and it turned out to be very hard to do. Then Earl Boebert, do you know that name?

Yost: Yes.

Levitt: Where was he then? He was at Honeywell. In fact, he eventually got the contract to take PSOS further. Very smart guy. We talked to him and then he wrote a paper saying that you can't use capabilities to do multilevel security. It's sort of an abstract paper. We said oh my God, we screwed up. Well, we had hacks, but anyway. The thing is to combine discretion and inter access became hard. Then KSOS, remember the KSOS system? Kernelized Secure Operating System. Okay, yes. So Rich Feiertag, do you know that name at all?

Yost: Yes.

Levitt: So he joined us and we came up with ideas. Bit of a hack. How to do multilevel security and capabilities work together. Then he left SRI to go to, oh my goodness, Ford, I guess? Wasn't it Ford? In Palo Alto; that became Sytek, I think. I think I got the right order. Anyway.

Yost: That was a name that I meant to ask Peter about, Rich Fiertag. Can you talk about his background a bit and what he did?

Levitt: Rich Feiertag? Oh sure, a great guy, very underrated. So you probably heard of noninterference in security? Does that ring a bell? Okay. Well, Feiertag discovered it first. My name's on the paper but it was his idea. Essentially, the idea was that — and this was the heart of multilevel security because it was information. You didn't want information to flow from a high-level object to a low-level object. He came up with the idea essentially that if you have a sequence of operations, some at high levels and some at low levels, and then you remove the ones at high levels, it had no effect on the string at all and then you were multilevel secure. We came up with that idea and I think he implemented that. That's essentially the heart of KSOS. And then Goguen and Meseguer came along and they formalized it. They did a good job, but it was Feiertag's idea. And he had a lot of background in Multics. He worked with I guess it was Jerry Saltzer. He never finished his degree — we discussed that, poor guy — and then he really knew system design, okay? Much better than Peter and far better than me, so he sort of took over PSOS and essentially constructed the levels of it and essentially did a real time PSOS, too. We called it RTOS and Rich was the brains behind that. Just remarkable guy and very underrated.

Yost: During the PSOS years of the 1970s, was that pretty much dominating your time at SRI?

Levitt: No, I actually had a lot of energy in those days so I was still working on verification, I was working on testing, and there was SIFT. You know about fault-tolerant computing? Did Peter talk to you about SIFT?

Yost: Yes, but please give me your recollections on it?

[BREAK in interview]

Levitt: We still had been funded by fault-tolerant computing by NSA Langley research center. They were interested in fault-tolerant computing and a fellow named John Wensley — I don't know if Peter mentioned him. Very sharp guy, not a theoretician but just a great designer. SRI had loads of people like that. So he had the idea, and we wanted something very elegant in fault-tolerant computing so we got to the point, at that stage, where we weren't worried about unusual transistors failing because it wasn't a unit that you could replace; think about replacing an entire computer. So he had the idea that what we would try to build a computer that would be able to mask failures in a single computer, no matter where they are. Voting was still the way to do it then, majority voting. So he had the idea of SIFT, Software Implemented Fault-Tolerance. I don't know if you've heard of that?

Yost: Yes.

Levitt: Great. So that was his idea. I was part of it but far and away, he was the brains of it. The idea was that we'll run a computation for a while — think of a real time system. If computers would do it, it wouldn't be synchronized — we'll get to that in a second. And then you'd vote on the results, and there'd be a voter in each computer so if one of the voters failed, that's no problem if we are getting anything failing, and then you continue and eventually you have to deliver the results to some actuator or something like that. A very brilliant idea, remarkably elegant. We had a paper in the IEEE proceedings, I think. I never remember what year. It's probably late 1970s or so. And then, all of a sudden, we came up with the idea that unh-oh, we have to synchronize the clocks otherwise you've got drift and you won't be voting in the same cells. And then we came up with the idea that gave rise to Byzantine fault tolerance where clocks could drift. Clocks could lie, essentially. You try to synchronize but clocks exchange values, it could behave in a Byzantine way, in a very arbitrary way. So we came up with algorithms to do that but it was unh-oh, wait a minute; when you're exchanging values you could lie too. So that gave rise to the Byzantine Agreement. This gave rise to the papers on Byzantine fault tolerance, Byzantine clock synchronization, and those became levels in the system. So we did a bit around PSOS, put levels in the system, and then we came up with the idea that gee, the SIFT is basically an operating system that involves scheduling and exchanging values, maybe we could prove it. So that gave rise to a project where we were trying to prove SIFT. And that was somewhat successful. We didn't prove code, but we sort of proved the design. The code is a little more difficult to prove. So, anyway, that was something else that was done at SRI at the time, a remarkably influential project. I would say, offhand, maybe more influential than PSOS. That's my guess.

Yost: Can you talk about some of the things it influenced?

Levitt: Sure. It was the first paper on Byzantine fault tolerance and it gave rise to thousands of papers since then on that. And clock synchronization, it gave rise to papers on that. And on saying maybe we could prove simple operating systems, so it gave rise to the fact that maybe we could send SIFT to proving real time systems and things like that. And then what else did it give rise to? Oh, there's BILT. I forgot who did it. Bendix, I think. Yes, Bendix built the hardware for that. As of a couple of years ago it still existed in Langley but I think it's gone; somebody just discarded it, unfortunately. What else did it give rise to? Oh, the design of design proof. So you can think of — we initially started doing work on coding proving programs, proving code. But somebody said well, maybe it's more the design that has problems rather than the code. Code can have problems, too, but you just get the wrong design or the wrong design metaphor. Something like that. So with SIFT we actually proved the design. It had rules to describe the system and then properties about these rules. And that's actually much simpler because we were just dealing with logic. We were abstracting away from programming language, all the idiosyncrasies of a programming language, like pointers, which people still don't prove. I think that was extremely influential over the years.

Yost: And who funded SIFT?

Levitt: NSA Langley Research Center and a couple very — you probably don't know the name Ricky Butler, do you?

Yost: No.

Levitt: Okay. You probably don't know these names, but anyway, Rick Butler's still at Langley, still doing work on proof, and proving aircraft systems.

Yost: So it really was a combination of just knowing what might be useful in the defense and intelligence communities, but also sometimes they came to you with some type of request?

Levitt: That's right. In this case, NSA, Langley came to us because they wanted fault tolerance. They were working on aircraft systems, so it wasn't part of the space program but, of course, redundant, fault-tolerant computers became part of Space. So I'd say SIFT became very, very influential. People I talk to all the time say that was extremely influential, and John Wensley deserves the credit for it. And Leslie Lamport was involved in that, too. You probably know Leslie Lamport, right?

Yost: Know the name, yes—and his work on distributed systems. You were also the co-developer of the first symbolic evaluator for…?

Levitt: That's right. I did it with Bob Boyer and that was the idea that Dijkstra thought was the stupidest idea he ever saw. [laughs] I think it usually doesn't scale well, exactly.

Yost: Can you talk about that?

Levitt: Sure. This was a time when coverage became an important metric for testing. So the idea was have you tested every statement in the program? A couple things came out. How do you derive test data that would give you good coverage? Number one. So it actually gave rise to it. The idea was you essentially just walk your way through the program with symbolic values, and then you try to determine, have I tested all the paths? There's the define critical paths in the program, and then you find out what tests were impossible because if you came to a branch, only one branch would be possible. And so you essentially had dead code in the system, so this was a way of identifying dead code in the system. It was a way of just sort of walking through the program, keeping track of the symbolic values as you walk through. And then one idea was you had, when you get to the exit, you eyeball the values and say, is this what you'd expect? Well, the performance can be 10 pages long so that didn't work. But the idea of trying to generate test data that would take it through critical paths *does* work and I think people still use that in practice now. In fact, static analysis of code essentially draws upon that idea so I think it was pretty influential. There's a paper called SLEC, which again appeared at this 1975 conference, and I think was referenced a lot for a good number of years. Bob Boyer's the one who built it; he built the system. He's a much better coder than I am.

Yost: And are there other projects that you worked on that we haven't talked about at SRI?

Levitt: A little bit. Have you heard of systolic arrays? Does that ring a bell? This became the rage in, I guess, the 1980s and 1990s. H.T. Kung. The idea was essentially that you have very tiny computers on a chip, and a whole array of them. And you get parallelism, and you would have sort of minimal communication just along the grid, so you wouldn't have to have long

wires. A fellow named Bill Kautz at SRI had the idea of cellular logic in memory. He said that. That turned into influence, I think, the work on systolic arrays for a long time. I think people probably still build some specialized high speed computers that way. So we had the idea of saying the logic of a small amount of memory and then an array of these devices. So we just came up with ways we could solve different problems — graph theory, number theory — just sort of being able to map those problems onto these kind of arrays. So it's a way of getting parallelism. I think that the systolic arrays that took over and I think people built them. I don't know, I haven't checked lately to see what's happening with them. I'd say those are the main things at SRI.

Yost: One thing that you mentioned early on was collaboration between different labs and departments at SRI. I get the sense that that was perhaps more prevalent at an earlier stage than at universities. Can you talk about the collaborative environment at SRI in the 1960s and 1970s?

Levitt: Certainly within the lab we collaborated a lot. Across labs maybe less so, I think because the Artificial Intelligence Center did extremely well at SRI then. They gave rise to Shakey. You know Shakey? Okay, good. That was big. Essentially that was influenced by planning and vision. I'd say there wasn't much cooperation across the labs then. I said I worked with Doug Engelbart, but that was just of course, you know, it was an opportunity to do it. I don't think most of us did it then. I think we were pretty siloed then; still, unfortunately.

Yost: A great deal of collaboration within . . .

Levitt: Within a lab. Oh, extremely so within a lab. Exactly.

Yost: . . . and probably more than the emerging computer science departments across the country.

Levitt: I'd say definitely yes. That's definitely the case. They were emerging at that time. Yes.

Yost: In 1982, that was the year you became director . . .

Levitt: Oh that's right, yes.

Yost: Can you talk about that administrative role and did you enjoy a heavier administrative role? And can you talk about the vision that you had for the lab?

Levitt: Well the answer is no. [Laughs.] I took over for Jack Goldberg and eventually Pat Lincoln, who you probably know, has the job now.

Yost: I know the name.

Levitt: He's a remarkable person. He does it much better than I did. The job should have financial leadership, management leadership, I guess, and also some degree of technical leadership although people who do good — Les Lamport and Joe Goguen and John Rushby and Bob Boyer and Jay Morrison; they didn't need much technical leadership. Peter Neumann. They

could do fine without me. The financial leadership problem, I guess, is the problem. You have to bring a certain amount of money to keep people sold. That was a burden and a worry so I didn't enjoy that at all.

Yost: And in the role, who did you report to at SRI?

Levitt: Don Nielson, you know Don?

Yost: No.

Levitt: He was, I guess, director of computer science. He was one of the directors of the packet radio project, which gave rise to cell phones, I guess. Gave rise to the notion of bay stations, and radios communicating. It was DARPA funded, very influential. That was his job. But he'd gone on from that project to become director of the division, I guess it was then. But no, I didn't enjoy being director at all.

Yost: One of the big tasks was making sure that individual researchers were working enough on contracts . . .

Levitt: . . . and not on overhead. Yes, exactly. And people felt that they were entitled to their share of overhead, not more than their colleagues. I wouldn't say there was any downturn in funding. I can't remember. It might've been during the Reagan era, there might have been some downturn in the research area, but I don't know. Unnecessarily. I think research has always been

fairly well funded. It was before security became very important. So now the job's a little bit easier, because security's just so important now that these days it's easier to get funding than it was then. Network security was just coming into play, in those days — certainly pre-web. ARPANET was just taking off then, and NSFNET, which gave rise to a whole new set of security problems, of course.

Yost: Was there any planning that you did with regard to a vision for what areas would be the most influential in the future . . .

Levitt: Definitely. At that point, intrusion detection had just come into play. I guess, the Anderson Report. I forget which one, but you mentioned that. So that was becoming important and then, let's see, what else. Network security. I forgot one name in the book, you might remember it. The follow-on to the Orange Book, it spoke about network security. Oh darn. Red Book? No, it wasn't Red. I can't remember what it was. It's one of them rainbow series.

Yost: I don't remember the colors in association with their contents.

Levitt: [Laughs.] And then security management—I don't remember what color that was—was becoming important then. And then database security. Security was changing from just concentrating on prevention and operating systems and access control and multilevel security, to the whole range of activities that we're concerned with now. That was just coming into play. And I think we wanted to innovate in intrusion detection then. Dorothy Denning and I, we got a project. She and I wrote a proposal to the Navy that gave rise to IDES and NIDES, and then I left

shortly after it, just before it really blossomed. Teresa [Lunt], of course, took it over from Dorothy but Dorothy was the brains behind it, for sure.

Yost: Alongside the IDES project, SRI also did some influential work on secure databases.

Levitt: Yes. Again, I just sort of helped write the proposal, but it's Dorothy's work on SeaView, multilevel secure databases. That was her vision. I mean, I was her boss but it was her idea.

Yost: In the early to mid-1980s, what did you think about intrusion detection and where that research would go?

Levitt: Wow. Well, I sort of staked my career in the last 20 years on it. [Laughs.] I sensed it was going to be very useful and I guess the reason would be that well, if you can't prevent the attack then the idea would be that maybe it was easier to keep intrusion detections up to date and keep operating systems patched. Maybe I had the vision to barely see through that. Again, it's sort of thinking about rules; detection intrusion is based on rules rather than code. Maybe if you could just look at the rules and say are these rules correct? The rules that characterize an intrusion in a system. That way you get a better idea of potentially how the system works. In this case, it's detect all, detect as many security problems as possible. I guess this predated worms, we hadn't seen worms yet. Maybe viruses were just coming about then, so make detection intrusion detect viruses. And then maybe she was thinking about hardware intrusion detection. But intrusion detection was an important area, and I think it still is.

Yost: One thing in interviewing Roger Schell, one of the things that came across is that high assurance is everything to him. At least that was the message with Roger as I understood it. Was there a divide in the computer security community between the researchers that were seeking the mathematically proven secure systems, thinking well, if you have that, there's no need for intrusion detection?

Levitt: Yes, that's right.

Yost: Were some researchers coming down in the camp of let's focus on, you know, secure kernels rather than intrusion detection**?**, and others saying that there's going to be a hole in these supposed secure kernel with intruders getting in so intrusion detection is essential?

Levitt: That's a very good question. I'd say that intrusion detection might've been partly focused on the insider problem in those days, insiders exceeding their privileges. I think Roger recognized this. Of course, he deserves a lot of credit from the Multics days and other things, but what he might not have told you is that I think formal methods would've died without Roger. I don't know if he would've told you that; he's modest. But anyway, this was in the 1980s, late 1970s or early 1980s when he was at NSA. So he recognized that we needed high assurance, that we needed proof, and he staked a lot on formal methods. He funded the work at the University of Texas, he funded work at SRI that eventually gave rise to PVS verification system, which is work that is still going on now, and other places, too. Oh and David Musser, you might not know. He's unrecognized. That gave rise to — oh, I forgot the name of her company, Debbie Cooper, where was she? She was president of the IEEE Computer Society — She'd been at

SDC. So Roger's sure that formal methods was going to be important and we needed mechanical verifiers in the late 1970s and early 1980s, and this was driven by high assurance. He deserves a lot of credit for it. Steve Walker, who was at NSA — what was that group? They go by letters. So this was "C" group, I think, and these people were involved with that. Ann Marmor-Squires. And he funded work on formal methods out of that work. Major investments. I think formal methods would have suffered a major slowdown if it wasn't for him. Anyway, so high assurance. I would say that initially, intrusion detection for the insider problem, the users exceeding their privileges. I think that that's essentially what it was. And I think it's exploded since then because we don't have high assurance.

Yost: And was it more the intelligence agencies than the DoD?

Levitt: At NSA, for sure.

Yost: CIA?

Levitt: It's hard to say. I sort of lost track of the CIA after I worked on formal methods. They fund us a bit here to look at intrusion detection. DoD funded work on, well DARPA, of course. When Teresa was at DARPA, there was a major program on intrusion detection.

Yost: And Becky Bace.

Levitt: When Becky Base was at NSA, sure. Exactly. She was very influential. In fact, she has a book on intrusion detection that's quite good. It's general, but there's a lot of insights in it.

Yost: I assume that you started going to the IEEE Security and Privacy Security Symposium from the early days forward?

Levitt: I haven't attended every one of them.

Yost: Did you attend the first one or other early ones?

Levitt: Early ones, yes.

Yost: Can you talk about that event and the atmosphere?

Levitt: It's gone up and down. Now, it's on a rise. Initially, it was very popular because it was the only security conference, and very high quality, and a fairly critical environment, which was good. The reviewers paid very close attention to the papers, and extremely influential papers came out of that. I guess, it hit a low point because maybe it was too theoretical for a while, not enough system-oriented; but I think it's come back now, for sure. It's remarkable, I mean, it's 33 years now.

Yost: Did you also attend the National Computer Security Conference regularly?

Levitt: Oh yes, I regularly attend that one because we've had a lot of funding with NSA over the years so I attended almost all of those meetings.

Yost: Can you compare and contrast those two meetings?

Levitt: In terms of academic quality there's no comparison; Oakland was far superior. But Oakland's a little weak on papers that dealt with systems and people that deal with policy. So I think the national conference is much better in that area. Huge audiences. Again, it's like the joint computer conference. There are probably 5,000 that attend those conferences.

Yost: I did not realize they were that large.

Levitt: Oh, huge number. The main site was Baltimore — somewhere in the inner harbor — the Baltimore convention center. Just absolutely remarkable, and I think NSA is the major force behind it.

Yost: Probably many people in government that aren't doing research just wanted to hear what was going on.

Levitt: Exactly.

Yost: What did you think of the National Computer Security Center and the goals with criteria setting and certification, The Orange Book, and what resulted in the published TCSEC standards?

Levitt: I mentioned before, Roger Schell. I think that formal methods would've lost 10 years if it hadn't been for the match. Because I think that focusing enough on formalities, I do see the need for it because without formality in understanding what you're doing and what securities you want, you won't get them. The question would be maybe The Orange Book was too dogmatic, and maybe these — I can't remember all these levels [laughs] — oh my goodness, going from verify to just any level, I'm drawing a blank. Senior moment. I thought that was a bit overdone. But they funded a lot of influential systems. KSOS came out of that effort, and then of course, PSOS would have some verification. I guess work on testing and building formal verification systems. And then, of course, they had the sense to move into networks and security management when they saw those were problems. Those reports are still very good. There's a lot of detail in them and a lot of good insights in those reports.

Yost: One of the big goals with The Orange Book, I think, in part goes right back to the Ware Report. Willis Ware articulated at the end of that report the importance of partnership with industry, that keeping research open and evolving with industry, that that would be absolutely critical to addressing the computer security problem. And obviously, the idea with The Orange Book, in part, was to inspire or incentivize industry to build systems that once they attained certification level, would be valuable to them, as generators of revenue and profits, but that didn't play out all that well. There were some systems that were developed and got B2.

Levitt: B2, yes, that's right. Now I remember.

Yost: And Roger worked on A1 systems.

Levitt: Had a couple of problems, that's right.

Yost: The Orange Book had desired effects concerning what you said about Roger being a force in keeping formal methods alive, I definitely see that. But there was also a goal for industry to really embrace strong security, and in that sense, The Orange Book had perhaps more intellectual impact than practical impact.

Levitt: That's true. It gave rise to a system, SCOMP, I guess. KSOS. But none became commercial, in part because maybe multilevel security wasn't seen to be a be-all to end-all. Security's a different story now, isn't it? I remember we tried to convince — well, "we" — a lot of people tried to convince industry that having compartments is what you needed. And yet, sort of partly what we have now. Obviously, access control is a big thing now no matter what kind of devices you have. So I think in that sense, it gave rise to awareness of this, in practice. But the public/private project, I don't know. It's funny, it's still an issue now. We can talk about that in due course, but I was involved with people who were in line to become cyber czars but didn't make it: Melissa Hathaway and Frank Kramer. You probably don't know those people, do you? But anyway, they were in line to be cyber czar, very recent history. They preach public/private/academic partnership. It's worked out in some ways. Clearly with Stanford it

works out, and maybe Berkeley, this academic and private sponsorship. And one guy who's very underrated — what's his name? I think it's Robert Rodriguez. He used to be with secret service. He's preaching that; he runs conferences on that with Doug Maughan. You know Doug Maughan?

Yost: No, I don't.

Levitt: He's at Department of Homeland Security. He heads the security program at DHS. So he has the security program at DHS and set up his own empire there. He's very good at his job, and he's preaching public/private sponsorship, much more so than NSA can. We'll talk about NSA in due course. But that didn't happen due to The Orange Book and maybe because multilevel security wasn't the cat's meow, and maybe because formal methods wasn't the right focal point, and then networking happened, too. All these things happened. NSA's mission probably changed, too.

Yost: At the 1974 IBM SHARE meeting, it's the first time that IBM really presented access control to their community of institutional and corporate customers. And out of that meeting, a project that had come out of IBM Research gets developed into a product that becomes RACF.

Levitt: My gosh, that's right. Oh my goodness. Wow.

Yost: As an outsider at that point no longer being at IBM but at SRI, how did you see RACF, in terms of was this robust enough a product or was it not really advancing the cause much?

Levitt: It was a good proof of concept. I kind of lost track of what happened to RACF, though.

Yost: It went through many different versions and it's still a product today.

Levitt: I'll bet the Federal System Division embraced it. And that helped because I think they did a lot of work with the federal government so probably influenced the world more than SCOMP did.

Yost: Its competitor was a product called ACF2 and a lot of major U.S. corporations, such as GM and others, either adopted one or the other. But I think it was rated in the early — not the very early years — but by the mid- to late 1980s, it was rated C1, tried for C2 Not a very high level, within The Orange Book security certification system.

Levitt: Right. But they recognized the need for access control as a big deal then and the separation of concerns.

Yost: In the years in the first half of the 1980s, when you were director, what research were you also actively involved in?

Levitt: So I guess the main thing was formal methods, then. This was when NSA still had a strong interest in formal methods. I guess Roger had departed, Steve Walker departed TIS, Ann Marmor-Squires, I think — do you know her name? I guess Hilda might have departed then. It

was Ann Marmor-Squires and Martha Branstad, who was Denny Branstad's wife. So they took over and they still wanted to embrace formal methods. So that was a major push then at SRI. I guess I was manager then, but at that point we realized the previous systems were pretty ad hoc; they weren't fast enough. I guess it was Rob Shostak, who was part of the Byzantine agreement, and Richard Schwartz, and John Rushby had just joined us. So they built the PVS verification system, that was the main thing. I was the manager and they were pushing that. And then there was — I guess that was the main influence there that came out, from what I remember of the early days. They were still working on verifying the operating system, at the time, designing operating systems. I think that work had an influence. Also coming into play then was the notion of changing verification — I don't know how to say this — seminal verification procedures, okay? Certain things you can do fast and automatic, and certain things you can't in verification. Certain things you need human input, some things you let it go. So like for example, solving linear inequalities. It's just a linear inequality; it's something that a program can do without any human input. So it's writing a program to do that. They're called system procedures. So that was coming into play, then, and it became part of this PVS verification system. Yes, I'm still a great fan of formal methods. In fact, Robert Boyer, who was at SRI then said, we may never get a secure computer — that was pre-network days — but if we do, then it's verified. And if they believe that at that time.

Yost: Steve Walker, who you mentioned, is someone I interviewed. He was at NSA, then DARPA, then industry, launching TIS. While at DARPA he partnered with NBS in building the community. When he left to form TIS, he starts the first computer security contracting company that grows large. I think it grew to, at its height when he was running it, to over 300 people. Was

TIS, between its origins in 1982 and the half decade that followed, operating, in part, in the same space as SRI, were they a competitor to SRI?

Levitt: We worked together on some projects. We competed for grants, for sure, and they had good people. They attracted a lot of people from NSA. I guess they could pay them more, maybe, and who knows? Who knows what else? Very good people. Bret Hartman, I don't know if that rings a bell. Ted Taylor, Bill Arbaugh, I think was with them.

Yost: Yes, they had quite a team, I believe Marv Schaefer for a while.

Levitt: Marv Schaefer, of course; I forgot about Marv. Very influential.

Yost: Steve Lipner was with them.

Levitt: Steve Lipner, that's right.

Yost: David Bell was with them.

Levitt: Wow. Yes, that's right. I forgot about him.

Yost: Some highly accomplished people.

Levitt: Yes. They competed with SRI; sure, friendly competition.

Yost: There was kind of the SRI team in computer security, there was also a TIS one.

Levitt: Yes. I think Rich Feiertag eventually went there. He left SRI to go to Ford, but then he went to TIS. And I had a number of my students, when I came here, became some of the TIS successes. I guess McAfee, maybe. McAfee bought out Sytek. Sytek? I can't remember. I'm losing track of the whole chain. But eventually McAfee bought them out, and then Sparta, and then I sort of lost track of them. But I think they essentially made security respectable and it was sort of a good bridge between the funding agencies and the private sector. They did lots of very good things. Steve Walker was just fantastic.

Yost: Before we move on to your transition to UC-Davis, are there any important areas that we've missed in computer security research that you either led or were a part of at SRI?

Levitt: Let's see. Maybe real-time securities. So following PSOS, we designed something called the RTOS, Real Time Operating System. Richard Feiertag was the brains behind that. This is an idea that if you had a secure; that in real time; well, first of all, you had to have information flow because of scheduling, you could get into covert channels. And the fact that, again, that scheduling became important because scheduling wasn't important in PSOS as far as security was concerned. That, I guess gave us some handle. I'm sure we weren't the first to think about denial of service because essentially, the schedule was important to assure that the task got scheduled on time and if the scheduler got somehow tricked into doing something wrong then you would get denial of service. So I think that was vital. We didn't do that much work on

network security in those days. It predated the internet worm. [Laughs.] And not much work on viruses. I guess I mentioned intrusion detection. I was the leader of the group with Dorothy Denning did the initial work on that and then Teresa Lunt joined us so I think that gave rise to that. Overall, the SRI experience was fun, a unique organization. You probably have some appreciation for what SRI does.

Yost: Definitely. Was IDES the first intrusion detection system that properly — I know, of course, it's part of the name — should be classified as an "expert system"?

Levitt: Probably, although people had rules. The rules are basically to just capture the signature of attacks. And other systems before that did that? Maybe not too many. I think he might have been the first to actually start thinking about rules to characterize the signatures. But I think IDES has really shown through its work in anomaly detection. That was very clever work. Building a profile and then detecting activity that wasn't consistent with the profile. Aging the profile. I think that that work has still withstood the test of time. They still teach it. So IDES, I think, was more important for its work on anomaly detection than for expert system. In fact, initially, we had it here building an expert system and my boss, Jack Goldberg, at the time, said why don't we call it an intrusion detection expert system. That's how the name IDES came about.

Yost: Can you tell me about the transition? What led you to leave SRI and come to UC Davis?

Levitt: Probably management [responsibilities], and didn't enjoy [it]; probably wasn't very good at it. And second, I recognized that — so this was the mid-1980s — and so Peter Denning wrote a very influential note in the CCM, quote, "we're eating our own seed corn." Does that ring a bell?

Yost: Vaguely.

Levitt: Okay, anyway.

Yost: I've read a lot of what Peter's written but not that note, though seem to vaguely recall references to it.

Levitt: At that time, IBM Yorktown and Bell Labs could essentially absorb all the Ph.D.s in computer science. I don't know how many came out, maybe 100.

Yost: I remember that, on the supply crisis. I didn't remember the name of the article but I remember something about that.

Levitt: The article was, "We're eating our own seed corn," and essentially said okay, all these young Ph.D.s — well, not only these young Ph.D.s from universities — they were going there because they paid more. So in those days you could probably get paid, I don't know, I'm guessing $100,000 a year maybe to go to work for IBM or Bell Labs. Universities were paying half that for full professors. I sensed that wasn't going to last. And so I figured I want to go to a

university, which actually I always wanted to do, so it's now or never. Five years from now it would be impossible. And I was right. So I came here and I sensed that maybe I could teach computer security and set up a lab here, get funding at that time. At that time, Davis had trouble, as most universities did, even to get people to interview for a job. People didn't want to go to universities. Stanford University could attract people, and M.I.T., but not sort of the second tier schools.

Yost: Higher than normal salaries there at the elite schools like MIT, and at Stanford, and often they were providing major housing subsidies for faculty- housing.

Levitt: Exactly. And then somebody who was on my exam committee at NYU, Mohammad Ghausi, was staying here and I said okay. So I approached him and I got the position.

Yost: Can you describe the computer science department when you arrived?

Levitt: We had seven faculty. We had 400 undergraduate majors. Fewer graduate students than we have now, maybe 50 grad students, but a huge number of undergrads because computers were getting popular then.

Yost: Heavy teaching loads.

Levitt: Exactly. And we had to teach so many different classes because with seven faculty . . . A lot of different classes to cover: operating systems, programming languages, software

engineering, artificial intelligence, computer architecture, discrete mathematics. I taught all those classes in my years here because we had to do it in order to cover those classes. And they were large, but it was fun. [I] got to learn a lot because there was a lot I didn't know, because I didn't take any computer science. I never took a computer science class in my life so I got to learn a lot and it was fun. It was very enjoyable and I set up the computer science lab, a computer security lab, at the time.

Yost: And when you taught at Stanford and also at NYU, and were back in New York, that was EE, correct?

Levitt: It was the EE, yes. No, at Stanford, I guess it was computer science. No, I'm sorry, it was when I taught error correcting codes, it was double E. When I taught a class in computer architecture at Stanford, it was computer science. No, it was still EE. I'm sorry, you're absolutely right. NYU was certainly EE. And we just had a very fledgling department. It was mostly math oriented at the Courant. Okay, that's it exactly.

Yost: Early in your teaching here, did you teach any courses specifically on computer security?

Levitt: I started a class in computer security here. Sure, and that was fun. It was based a lot on Dorothy Denning's book so I thank her for having that book. It would've been impossible to teach that class if it wasn't for that book.

Yost: How did students respond? I mean, did that attract a lot of students?

Levitt: Yes, because, I guess they sensed this was an important area so it attracted an awful lot of students. And I combined that with formal methods so we ended up — I was just desperate to try to get funding to support these students. And at that time, I met Becky Bace. She was at NSA and she started the university research — well, not started, but she helped spearhead the university research program at NSA. And DARPA took an interest in computer security. So I spent a good chunk of my time writing proposals to get funding for the graduate students.

Yost: When you arrived, can you describe the graduate program at UC-Davis?

Levitt: Yes. So we didn't have much funding so I think, if I remember, maybe we had trouble funding a level that would have involved just a couple hundred thousand dollars a year then. Still too cheap, but still it wasn't enough. So gradually we started with maybe 50 students at the time. Most of them were TAs. They were good students, very eager, but the graduate program was just starting then. We probably had graduated very few Ph.D.s at that time.

[INTERRUPTION]

Levitt: So, we just thought we haven't produced many Ph.D.s by then. Oh, and we were tied to links to Livermore, at the time because Livermore was actually computer science, primarily for their supercomputer efforts that were going on. So actually, most of the activity with graduate students was centered at Livermore at the time. It's something called Hertz Hall; it's outside the fence.

Yost: So some of the students worked at Livermore?

Levitt: Yes, many of them worked at Livermore. Exactly. It'd come up, and I didn't do it, but some of us went down to Livermore to teach classes there.

Yost: When you arrived were you the only faculty member focused on computer security?

Levitt: Oh yes. Definitely.

Yost: I assumed that.

Levitt: Yes.

Yost: What was the origin of the center?

Levitt: Here?

Yost: Yes.

Levitt: Well let's see. I worked with some of my colleagues. Ron Olsson was an expert in programming languages and operating systems. And he knew about security a bit from designing programming languages. So I guess it was the two of us together, primarily. Then a fellow

named Biswanath Mukherjee, I think he came in [and] he set up the network lab. We worked together. Then we had a couple good students. One of them was named Todd Heberlein. Actually, he built the first network intrusion detection system. He was a student of Mukherjee's and myself. We just called it NSM [Network Security Monitor] and it was the first packet sniffer. Remarkably, it sounds so easy now, right? But essentially, he looked for activity, did anomaly detection, similar to what IDES does but at the network level. So I guess that was the first of those systems and he deserves credit for that. So he was our student and there's a connection between networks and security, so that's how the networks and security labs sort of came together. They went separately, too. And then, oh, viruses became important then too. That family of floppy disks, of course. How to detect viruses, we think of intrusion detection as a way of doing it. What operating system features you need to detect it. Various things that came out of it like on the side of the ability to detect computer viruses. Could we prove that a computer program doesn't contain a virus? So we came up with heuristics but you couldn't prove it because of undecidability. So there's a lot of that came out of that work. Fred Cohen, of course, was influential in the area of computer viruses. So all that came together. We were funded by Livermore, believe it or not. Not believe it or not; we *were* funded by Livermore to do work in that area.

Yost: And when did you launch the center?

Levitt: Very soon after I got here, maybe within a year after I got here. It seemed like the right thing to do and universities want centers, of course. Through a center, then you can get funding.

Set up labs, we called it security lab then, and I guess it still exists. Of course, then Matt joined us, and others, over time.

Yost: Were there other university centers in computer security that you saw?

Levitt: Wow. So we predated Purdue, I think, in that area. Probably not, I'd say. I mean, other people were dabbling in security. I'm sure we weren't the only ones. For sure I can't say by any means that we were. Virgil Gligor, at Maryland, had an activity. In fact, I think Bill Arbaugh was his student. Stanford, probably not that much. Berkeley not that much. Yes, so we might've been the first. There were groups doing cryptography then, of course. I mean, Marty Hellman and others at Stanford were doing cryptography as well as the M.I.T. group, but I'd say we were among the first to look at system security. Since then, of course, a lot of people are doing it.

Yost: You mentioned the center facilitated sponsored research, what centers are designed to do.

Levitt: Yes.

Yost: Can you talk about what it meant — and this is obviously related — for graduate education, opportunities for graduate students?

Levitt: Yes. We certainly had a whole lab for graduate students to do work, undergraduates, because we've always had very good undergraduates.

Yost: That's a shift from primarily TAs to RAs. . .

Levitt: The research, RAs. Exactly. That was certainly part of it. And then we tried to think about other areas. We involved some people who knew artificial intelligence, because you can think of security as being an artificial intelligence problem or a game between the attackers and defenders. We thought about all those things then, and did some work in those areas. But [the lab was] to give a place for graduate students to hang their hats, okay? And jobs were good. The job prospects were very good; and the university's too. So we graduated some students who went to — let's see, do you know Deb Frincke? She's Deputy of R [Research] at NSA. So she was our student here in security. She went to Idaho, and then she became a manager at PNNL, Pacific Northwest National Lab, and now she's at R. So she came out of this. And students are at schools: Brigham Young, the University of Idaho, Portland State. I'd say yes, we moved some students into university positions. Lately, they mostly go to industry.

Yost: I saw one of your students is at SRI.

Levitt: Steven Cheung! He's very good; an awfully good student. Yes, in fact, his thesis did some influential work — a *lot* of influential work — on intruder detection SCADA [Supervisory Control and Data Acquisition] systems. He's done some very influential work on that.

Yost: In your early years at Davis, can you talk about how your research interests evolved?

Levitt: Yes. Teaching was a major drain at the time. [Laughs.] I don't know how I did it. I probably couldn't do it now, and I don't remember how exactly. Most of us volunteered. We needed people to teach classes and I guess this is the way we'd learn something. I dabbled in some of these areas before, okay? So I knew a bit about AI because I did work on verification and theorem proving and then computer architecture, and got interested in logic design with Doug Engelbart. And then what else did I teach? Discrete math, which is something we should all be able to do. Programming languages and operating systems, well I know about PSOS, but I didn't know a lot about operating systems. But I knew enough to get me halfway through the class. So a lot of it was teaching and extending our graduate program because we needed to have more classes in order to have students. So they did computer security, had formal methods classes, operating systems, graduate programming languages classes. So all that came about. And doing a graduate class, you can steer it around what you know. I have a fair amount of freedom, which is good and bad for the students. They get somebody who maybe knows about this area but they don't see everything about the area. And Livermore helped us an awful lot in getting funding. So they funded initial work on this Network Security Monitor, which is the first network intrusion detection system. They funded that. And then people at the Air Force funded— this guy named Tim Grants, who you probably don't know, but he set up the security group GRNC, his security group at NIST. Then he was at Kelley Air Force Base, Air Force Information Walker Center I think. So he funded work on a distributed intrusion detection system. Do you know Steve Smaha, do you know that name?

Yost: Yes, from Becky's book.

Levitt: He did the Haystack. So we teamed with him. So you could say that's an academic/government/industry cooperation. Smaha was the brains behind it, along with my student. The first distributed intrusion detection system research — get reports from various places and fuse them together, and so we did that. And then we got work from DARPA for something called GrIDS [Graph-based Intrusion Detection System], which is a large scale intrusion detection system. Slowly but surely we went. Teresa Lunt was part of that. She wasn't the program manager for the first guy that funded it, but she took over that program. So I guess a large number of efforts came out of this, mostly intrusion detection. We sort of centered around that. We did some work on viruses, some work on worms, but mostly intrusion detection systems and formal methods, too, but we haven't done much work on formal methods lately. I guess my SRI experience could help because somebody had to write proposals. How to listen to program managers say what they wanted. In fact, you know Dick Kemmerer? You must know him, don't you? Richard Kemmerer.

Yost: Yes, the name, and some of his articles. Never met him. I hope to interview him on this project.

Levitt: He's a good guy. We joke around a lot. He's a good guy. He said to me well, it's not fair that assistant professors have to compete with you because you know how to write proposals for funding.

Yost: Definitely something where practice and experience builds skills.

Levitt: Exactly, and I owe it all to SRI. Believe it wouldn't have happened without me being at SRI and working with people who knew how to write proposals.

Yost: Was there a real environment in your early years of more senior people providing guidance on proposals?

Levitt: Absolutely.

Yost: A lot of mentoring.

Levitt: Yes, a lot of mentoring. SRI was wonderful for that. Again, it's unique in that area because you work for a place like Yorktown, you don't have to write proposals. You have to get somebody to appreciate your ideas, but writing proposals is different.

Yost: Places like SRI, RAND . . .

Levitt: SRI, RAND, Lincoln Lab, I guess.

Yost: . . . where you learn those skills.

Levitt: These places are almost unique, in terms of research. A lot of organizations write proposals, of course.

Yost: And even though SRI and RAND have some commonalities, they seem like very different institutions.

Levitt: They're very different. Well, RAND is sort of captive, I guess, in some ways to the Air Force. SRI's captive of nobody, which means they don't have any benefactor.

Yost: Well, it certainly started that way, that's shifted.

Levitt: It has shifted, that's true. Yes.

Yost: The first 20 or 15 years . . .RAND was heavily Air Force research focused.

Levitt: The second Lincoln Lab. Lincoln Lab, I guess, was captive to the Air Force, too.

Yost: . . . a whole lot of social science research at RAND in recent years.

Levitt: You're absolutely right, I think. But that's true around the country. No question about that.

Yost: You also listed this under research categories and published a number of articles about argumentation with application to security. Can you talk about that?

Levitt: Yes. A very smart guy at CUNY, City University of New York, Simon Parsons and his wife, Elizabeth Sklar sort of own the field of argumentation. The theory is that you can think of a number of parties sort of arguing with each other and you want to present their arguments in such a way that the other party can accept them. You think this has been a dialog or, I don't know about a multi-dialog; a trialog? Whatever it is. A multilog going on until maybe you convince some of the parties so that other parties cease to argue with you. So you can think of coming up with arguments and the arguments would have preconditions, and you can rebut the preconditions, you can rebut what they produce, and eventually decide that gee, you know; and the reason is that you can't come up with an expert system or an artificial intelligence that can completely decide this. It's a way of sort of combining logic with human reasoning and human persuasion point of view. So we're working on that from the standpoint of security administration, so at some point, I think we'll have time for talking about what the major problems are in computer security space. Again, I get that from my NSF days. But anyway, this is the problem we're working on now to try to come up with arguments for a security administrator for what to do, what actions to take, whether to patch or not patch, whether to block traffic or not block traffic, whether to change my firewalls. The idea would be to treat this as an argument, possibly with other people, because a security administrator has to argue with administrators at a university about what do I want to allow from the standpoint of security? Is security more important than privacy? So you can imagine getting into that kind of discussion. It's an interesting area and we're getting funding from NSF to do that now. And we also get funding from the Army Research Lab to look at this from the standpoint of deciding in a battlefield situation. If I'm a soldier in the field, what should I do now? Maybe I'm getting orders and maybe suggestions from above, maybe I'm seeing certain things there, and I have to make

decisions on my own. So we think of this as being a formalization, a characterization of what he has to do.

Yost: What other area of security research are you doing at the center, what about social informatics, can you tell me about that?

Levitt: I'm not doing too much on that. I have colleagues who are. I have a colleague who is doing work in that. Well, social networks give you ways for attackers to communicate with each other or attackers to learn which part of a system to attack — which users are vulnerable based on their social informatics, their social connections. So I think social informatics is a way that attackers will be trying to get into systems, and the way defenders can do it, too. The way they find who my friends are I can depend on to help defend this system. These are all potentially good ideas, I think. Not fully understood yet, by any means.

Yost: Have many malicious hackers gone into the field of intrusion detection?

Levitt: Well, Kevin Mitnick for sure.

Yost: That's one example but I was wondering is that a broad phenomenon?

Levitt: I don't think it's unusual. I don't know a lot of people. So, Becky's the one — you probably talked to Becky. She had a project at NSA by the name of Project Slammer. Did you talk about that?

Yost: Yes.

Levitt: She's really the expert on that. And Donn Parker, too. Have you talked to Donn Parker?

Yost: Yes. In fact, I interviewed Donn well before this project. Donn's actually an important force that influenced us to do this project.

Levitt: Oh, okay. So it's funny about Donn because he said he interviewed a lot of hackers and he's influencing them in ways both positively and negatively. So when I first came here, I was involving graduate students, he said that's a big mistake. You should not involve uncleared people in doing work on computer security. Well, the world's changed. I'm sure Donn's changed too. At that time, we tried to defend against attacks. Well, you have to dream up attacks and we can talk about this a little bit later, but attacks on the power grids, for example. I don't know what to attack on the power grid. We have to sort of dream them up. He felt that's just a computer virus. It's a big mistake to have uncleared people dreaming up attacks. You'll get into a lot of trouble for this. Well, I guess I haven't. [Laughs.] This is 30 years ago he said this to me. Almost 30 years, well, 28 years ago.

Yost: In 2005, you became the program manager for an NSF research program for computer security.

Levitt: Yes, replacing Carl Landwehr.

Yost: Can you discuss both the decision to take on that role as well as your view of the landscape of the broad fielder of computer security and what you wanted to do with that program?

Levitt: In one way, it was sort of personal. My wife had recently passed away and I figured okay, why not to try something different. Somebody here, my dean at the time, a fellow named Alman Lau said to me — I guess he heard about my opportunity — said why don't you do that? In his words, well you can graduate another 20 Ph.D. students but you can have much more influence on the community by going to Washington. And of course he was right. But it's hard to say that I felt like I influenced the field. I fear it would be a little pretentious. I mean, we all influence; *you're* influencing the field. It's hard to have a major influence. It's such a huge field right now. But I felt maybe I could. Maybe I could get people to work more closely together on it; maybe I could help steer. In those days, what was I doing? Largely intrusion detection. Maybe combining different fields together. For example, intrusion detection and formal methods, and maybe intrusion detection and AI, maybe I could have some influence there. Also perhaps on computer security education because I knew NSF had a major interest in that area. So, scholarship service programs. So again, it's partly personal and partly I didn't quite know what I was getting into, but partly I can make this happen.

Yost: I'm sure the program had an influence on the number of computer security courses that were offered.

Levitt: Absolutely. I think the program has a major influence on classes on network security, for example. My colleague, Prasant Mohapatra, for example, concentrates on network security at the physical layer. So that's part of our curriculum now, and a part of curriculum at many universities. The way that computer science has gotten cubbyholed, siloed, whatever the word is — we can talk about that in due course — it's a major issue, a major problem I think that we have to face. It's probably the way NSF does its business, too, but we can talk about that in due course.

Yost: Are there particular projects that you funded in that half decade that really stand out to you? And can we talk about some of them?

Levitt: So, NSF tries to fund projects at different levels: small, medium, and large.

Yost: We're a small one.

Levitt: Yes. And there are small, and very small, too. So I tried to have the larger projects and the most influence. They're the ones where we tend to have the most influence, and they're the ones we tend to talk about more. First of all, we're funding, arguably, the best people in the area, and on topics that were — I tried to do things that I could talk about to Congress. I didn't talk to Congress very much or interact with them. Somehow you wanted to convince Congress, at least indirectly, that you were doing important things or you're changing the way things are done. So a couple of things, as I inherited some projects from Carl — I managed them — that I think I had a

fair amount of influence. One was the accurate project in voting. Did Peter talk about that, didn't he?

Yost: Yes.

Levitt: So I felt that I had a lot of influence on that. Originally, they were trying to set up a lot of technology so that we have secure voting machines. Well, in the end I guess we've gone backwards. You live in Minnesota, don't you?

Yost: Yes.

Levitt: I don't know what they use there, but now we use paper ballots here.

Yost: As we do in Minnesota.

Levitt: That's remarkable. I think if I actually had the insolence, if somebody tried to — I shouldn't use discredit — but tried to say there were problems with Diebold, and no, you can't trust them. That was one direction.

Yost: Very important, actually.

Levitt: I think so. I think you're right. But maybe we don't need 10 hours of voting. I tried to also influence them to think could we ever do internet voting? Peter says no. Okay. But I worked

a bit with David Chaum. You know David? Okay. He's a cryptographer. Where is he, Santa Barbara? I've sort of lost track of him. Remarkably smart guy and I tried to persuade him, could we ever do internet voting? He's a cryptographer, and he said yes we can. With the right cryptography we can make it work. You can even be sure your vote was counted. You have no idea where Diebold's counting now, right? How could you know?

Yost: Just assume it was.

Levitt: You assume. Well, he had ideas in cryptography and said well, you could. You could see an encrypted version of your ballot in the newspaper of how you voted, and you have the private key, so essentially you can see ah, that's my vote; that's exactly who I voted for. And there's no way it could be changed. You could do a whole chain of custody using cryptography, and we could do this over the internet, even with insecure systems. Peter Neumann says no. [Laughs.] Okay, we disagree about that.

Yost: People say if you take voting away from the polling place, someone else forcing —

Levitt: Exactly. So Carl Landwehr mentioned that to me on numerous occasions, and Brian Randell. But how many people do absentee ballots now, right?

Yost: Yes, that's true.

Levitt: In fact, I think Oregon is almost all absentee ballots.

Yost: I guess I don't think about that, that much, because it's very low in Minnesota.

Levitt: Okay, very high in Oregon. Pretty high in California; at least 10 percent in California.

Yost: My parents live in Washington State. It's pretty high there.

Levitt: You're absolutely right. An answer could be maybe you can revoke your ballot at some point? There's cryptography that lets you do that. And you might have a certain amount of time, a day? I don't know how long you'd have to do that. Years? Probably not. Technology can handle that, okay? But you can be coerced into revoking your ballot, too, so; as we say in security, it's a cat and mouse game, for sure. But David Chaum persuaded me that we actually could solve the internet voting problem even with an insecure operating system, except for maybe denial of service. So maybe somebody could bring the internet down on election day. But, I mean, it's okay; you could bring it back up again. The internet's pretty resilient. So anyway, I funded that work for him to do that. It was a little bit separate from ACCURATE, but ACCURATE had a tremendous amount of influence. And I think I tried to convince several people in ACCURATE, why don't you try to prove the correctness of your voting machines? One of them was used to count, basically. I think they might've done it without me, but I think they tried to do that. So it was a very broad view of security and I think that was very successful. And then work that Bill Sanders did. You know Bill Sanders? He's at the University of Illinois Urbana-Champagne and he had a project that again Carl funded and I took it over shortly after that, on security for the power grid. So it was, I think, very influential. This is a very hard

problem, and it's a hard problem because I'm basically going to tell you what the vulnerabilities are? [Laughs.] Of course not. They're not going to tell graduate students from China what the vulnerabilities are. So you have to sort of guess and speculate what the problems would be. But over time, as you were saying before, government/private/academic relationships, you have to get to the point where you trust the academics. Some academics go to PG&E, for example, and they're trained in security. Maybe the utilities will trust us enough so that they'll tell us some of their problems. We'll sign nondisclosure agreements. We'll try to abstract out the important problems in security for the power grid. So I think those centers' project was very influential, in that regard. And let's see, oh yes. And then the work we funded, again Carl started but I took it over, Stefan Savage. You know that name? Vern Paxson? Okay, So Stefan Savage at the University of California San Diego and Vern Paxson's at Berkeley. So Vern Paxson built I'd guess, arguably the best intrusion detection system until now. It's called BRO, stands for Big Brother. Good name. [Laughs.] Remarkably good system. There's packet sniffing. It's fast because he uses high speed parallel networks to do it. It's flexible, but overall what this project did was — I can't say it succinctly — it started looking at security epidemiology. How can we be bio-inspired to think about pathogens? I'd say that didn't work out but what he did work out was computer crime. So they were the first to say okay, this was the day spam started coming in — botnets and the like. And then I would say the underground echo system started coming about. They were one of the first to recognize this and study it, looking at it in a principled way. By looking at packets, capturing packets and honeynets, they tracked the echo system down. They followed the trail of money. They did reverse engineering of the protocols that the underground economy uses to communicate from the botmaster to secondary units, to the individual bots to control them. And they started to infiltrate it. I guess we were worried about their kneecaps a bit.

It turns out the underground economy is doing absolutely fine. [Laughs.] And of course they're in touch with the FBI. I would say I monitored that work, okay? And then we monitored some work on the beginning of cloud computing too, because the mid-2000s were when clouds began to become prominent so we funded some work on that. So some successes there. Some places where we weren't successful: biometrics. I don't know, it's a hard area.

Yost: What do you see as the greatest vulnerabilities in computer security today and in the near future?

Levitt: Oh my goodness. That is a great question. I get asked the question from Congressional staffers a lot. Think of the story of the blind men and the elephant; everybody sees it differently. So if you ask, some people say insider problem, some people say zero day attack, some people attacks on critical infrastructure, some people say we have insecure systems, some people say the programming languages are bad, some people say ah, your grandmother, right? She clicks where she shouldn't be clicking, and that's what creates all these bots. So they all see it differently. What would I say? Oh, and some people say — and I was part of NSF at the time — we need a Manhattan Project in security. Wow. Yes, some people buy that. Oh, and some people say the Chinese, of course. Yes, right, the Chinese are going to eat our lunch. [Laughs.] In part because they have insiders on our system and in part because they have so many people who can find these vulnerabilities in systems that become zero day attacks. Some people say it's policy; we're not allowed to attack back. Well we can't attack back. We don't have provenance. We don't know who's attacking us. So what would I say is a major problem? At NSF we funded work in all those areas and have continued to do that. Wow. I guess it still boils down to vulnerabilities in

systems, vulnerabilities in code, I think. So now it's in browsers maybe more so than operating systems that are being exploited, and the like.

Yost: Are there particular browsers that are more vulnerable? Are there more attacks on Explorer than . . . ?

Levitt: Probably more attacks on Explorer. Just because it's more prominent. Typically, they're the same kinds of attacks we saw on operating systems. They're buffer overflows. They're race condition attacks. In some cases they're just attacks finding ways to escalate privileges and get foreign code running in the systems. In fact, Stuxnet exploits zero day vulnerability, and to get foreign code into the system. So in that sense, things haven't changed; I mean, the targets have changed. It's just that I think phones are better, for some reason. I think their design — they learned a little bit about separation. In fact, phones don't communicate directly with phones, which helps an awful lot and people somewhat control their privileges when you download apps. But insecure apps would then be a problem, for sure. But so far, we haven't seen attacks too much on the infrastructure of phones, you know, on iOS itself or Android. It's more on the apps and I don't know why. Maybe iOS and Android are better. They probably are, to some degree. But then you could think about Sensa networks, you could think about Sensa networks in the field. They could be attacked. Medical devices, ah, there's where people say those can be a source of attack so we actually funded work — you maybe heard mention of this — on Chinese pacemakers. That made the *Washington Post* when NSF funded it. We have to scold this guy a bit. That's not helpful. I won't tell you who it was. Okay, so privacy, of course. I mean, there's so many different areas and I could ask, should we have a Manhattan Project and I don't know

the answer to that. Something ground up. Maybe it makes some sense so we could have a base; oh, supply chain problem, okay? DARPA called that the security problem from hell because we're getting chips from China. We're getting phones from China, they could put a hardware Trojan horse inside the system. They could hide it. Yes, I mean you can possibly fund more work in that. Boy, that's a very hard problem. I would say designing something ground up, getting a security architecture we could all agree on. I don't see that happening. From ground up, you know, secure hardware, maybe making a tag based upon these old ideas of the B5000, and then a secure operating system build with a type-safe language so that we'd essentially eliminate all these peripheral attacks and make it simple so we can verify it. Sort of make it like a hypervisor. And then maybe having to support monitoring intrusion at all external levels, but that's my hammer so I have to be careful about that. So monitoring at all levels in the system so you could see if there's anything happening that you don't trust, anything at the kernel level because if you don't detect them, they're inside the system, and they survive rebooting.

Yost: What do you see as the most important lessons from the past in computer security that today's generation of computer security researchers are either ignoring or at least not paying enough attention to?

Levitt: Wow. That's fantastic. I'd say, security researchers now seem to be impatient. In part because maybe a lot of them have startup companies and I think they just forget principles. They don't have any strong driving principles in dividing of systems, like separation of privileges, or doing information flow. The formal methods don't exist. There's a tremendous amount of work in stack analysis cover to find vulnerabilities. Some very successful companies that I think came

out of NSF work, for sure, that Carl Landwehr funded and I funded. Some of those are successful companies. Microsoft, I think, does some of this. But people, I think, don't want to use this. I think what we should learn from is — okay, again it's one of my hammers, but some of these principles and formal methods. So microcode is now verified and I think Roger Schell deserves the credit for this, this supporting formal methods. Intel, I think, verifies every line of microcode before it could happen. Chips don't get recalled now for microcode problems, and I think it's because it's verified. And I think we should — I hate to say verify all the code in an operating system, that's going to be ludicrous — but at least have them driven by principles and some kind of formal methods. People just don't do that now.

Yost: And finally, are there any topics that I haven't brought up that you'd like to discuss?

Levitt: Let's see, well, computer security education. I think we have to, again, teach these principles. My students just don't want to — I don't know if they're bored by it or they're just impatient. They want to build, they want to do coding, and I think they try to do it too early. What else? I think, again, this idea of a reference security architecture. It's not completely my idea, by any means; it's part of what Peter Neumann's saying, part of the CRASH program. I think that makes a lot of sense. Try to go from the ground up. Again. The hard thing is you say well, what about legacy systems? Will Microsoft ever accept it? Okay, what about Windows? Well, there are ways around that. You can hypervise it, allow guest operating systems to run. Maybe we don't design Windows right away. Windows won't be around forever so I think a reference architecture is something that should get done. There are several people working on it now. I think mostly from Howie Shrobe's program at DARPA. I think I'd say that's clever. What

else haven't we just talked about? I think funding bigger projects I think is important because I'd say when I was at NSF, I guess a lot of mileage came out of these larger projects so I think that's important to do. And DARPA, I don't know what DARPA's doing in security these days. They're trying, okay? Doing lot's of work. They're taking on fully homomorphic encryption, which I think is a very important topic. One of the things against the reference architecture world, the field's changing too fast. We couldn't have anticipated fully homomorphic encryption 10 years ago, so [if you] had a system that wouldn't admit that and you had a problem, it might be a dinosaur.

Yost: The landscape of federal research, what portion is NSF's Trustworthy Computing (now SaTC) and DARPA, and how large is the federal research expenditure on computer security overall, do we have any idea?

[BREAK]

Levitt: So talking about expenditures. Boy, it's hard to say. Like take NSF. So NSF's program — that was Carl, me, and then Carl — spent $75 million and most of that money comes, believe it or not, DNI, Director of National Intelligence. Because I think they saw a need for it and they wanted to fund research, so I think they funnel money through NSF. That's public information, okay? Nothing's crooked about that. Then there's AFOSR, ONR, and ARO, the research arms of the defense department. I would guess they could be $10 million each on security, I'm guessing, on security, maybe a bit more. DARPA, God knows. [Laughs.] A lot of their work is dark. I don't know. I would guess they could well have several hundred million in security, I guess. I

don't know Butch Thompson's budget; I'm guessing $3 billion. I don't know but I always remember it as being half NSF's [budget]. NSF budget is $7.5 billion. And so total, probably spends — besides the program I had, secure and trustworthy computing — I'd guess it goes maybe another $15 billion or so. So NSF might have $125, $150 million in this. Since the cyber security R&D Acts — you probably know about those — they essentially mandate the NSF expend funds, invest money in security, and a certain amount. And I have no idea what NSA spends on security research, or the CIA, or the FBI. But there's something. You probably know about NITRD a bit, I guess? It's sort of — I forget what it stands for — but it's part of OSTP and it's a group of agencies that cooperate in trying to sort out security problems. So NSF is part of that and they try to avoid overlap in security funding, at least talk about what they're doing, they try to cooperate with DHS. DHS program must be $25 million in security research, I think. Doug Maughan heads that program. So it wouldn't surprise me if we weren't talking about, I don't know, at least $500 million in security research in the country, federal investments. Maybe more.

Yost: You mentioned a couple people who were passed over for cyber czar. Can you elaborate on that?

Levitt: Oh, yes. Ouch. I guess it's politics, of course. So Melissa Hathaway, she was at Booz Allen and then, I don't know if she was acting cyber czar, but anyway, she worked for DNI. Mitch McConnell is the Minority Leader in the Senate; another McConnell, maybe Mike McConnell. Anyway, so she worked for him and she did something, a 90-day review. You might have heard about that. I don't know. This was during, I guess, our President's early years. So she did that and we helped her with that at NSF. I guess she was in line to be the cyber czar. She

didn't get the position and next in line was somebody named Frank Kramer. I don't remember where he came from, something called the Atlantic Council. And then he didn't get it. Eventually, Howard Schmidt, I guess got it. I wasn't involved at all with Howard Schmidt. Anyway, I don't know what the office is doing in Washington, at all. I just don't know.

Yost: That was my next question.

Levitt: This certainly comes down through OSTP, through their present science advisor and I think that filters down to NSF and the like. They set certain priorities in terms of security but boy, they're vague, of course. It's hard to say. They're talking about things, and lots of studies that go on and kind of say "next steps." In fact, I'm involved with something now called 2025. So it's trying to think about what are security challenges in the year 2025. And that's something that's coming from OSTP, helping NSF deal with the challenges of that. So it's hard thinking about that. What are the challenges? What are the technologies that will help and hurt security problems? Will privacy be an issue? What are the applications that we're worrying about? You know people are thinking ahead. We're going to have world computing then, as an example. Huge privacy challenges and supply chain problems, too. Someone puts a Trojan horse in all the pacemakers in the world My God! What can that do to us?

Yost: Frightening to think of all the possible scenarios.

Levitt: It's frightening to think of all that, exactly. And you and I and the rest of the world that's involved think there's all sorts of bad things that can happen.

Yost: Thank you so much.

Levitt: I thoroughly enjoyed it. You asked very good questions.

Yost: This has been extremely helpful.

Levitt: Thank you, I thoroughly enjoyed it.