UNITED STATES DISTRICT COURT

CENTRAL DISTRICT OF CALIFORNIA

HONORABLE DAVID O. CARTER, JUDGE PRESIDING

\- - - - - - -

ECHOSTAR SATELLITE CORPORATION, et al., Plaintiffs,

vs.

NDS GROUP PLC, et al., Defendants.

No. SACV 03-950 DOC
Day 5, Volume III

REPORTER'S TRANSCRIPT OF PROCEEDINGS

Jury Trial

Santa Ana, California

Wednesday, April 16, 2008

Debbie Gale, CSR 9472, RPR
Federal Official Court Reporter
United States District Court
411 West 4th Street, Room 1-053
Santa Ana, California 92701
(714) 558-8141

EchoStar 2008-04-16 D5V3

APPEARANCES OF COUNSEL:

FOR PLAINTIFF ECHOSTAR SATELLITE CORPORATION, ET AL.:

T. WADE WELCH & ASSOCIATES
BY: CHAD M. HAGAN
CHRISTINE D. WILLETTS
WADE WELCH
Attorneys at Law
2401 Fountainview
Suite 700
Houston, Texas 77057
(713) 952-4334

FOR DEFENDANT NDS GROUP PLC, ET AL.:
O'MELVENY & MYERS
BY: DARIN W. SNYDER
DAVID R. EBERHART
Attorneys at Law
275 Embarcadero Center West
Suite 2600
San Francisco, California 94111
(415) 984-8700

-AND-

HOGAN & HARTSON
BY: RICHARD L. STONE
KENNETH D. KLEIN
Attorneys at Law
1999 Avenue of the Stars
Suite 1400
Los Angeles, California 90067
(310) 785-4600

ALSO PRESENT:

David Moskowitz
Dov Rubin

I N D E X

EXHIBITS

SANTA ANA, CALIFORNIA, WEDNESDAY, APRIL 16, 2008

Day 5, Volume III

(1:01 p.m.)

(Outside the presence of the jury.)

THE COURT: All right. We're on the record. All counsel are present.

I've gone back to tell Mr. Bender that we have a call in to Kathy Peel (phonetic) at Orange County Transportation Authority seeing if we can reschedule working with the County the date of his presentation until Monday afternoon.

Kathy Peel has not called back. I wanted to assure Mr. Bender that the Court had followed through with that effort. That's why I was back there for a moment. There's been no other conversation about the case, except I always take my clerk with me. And a note was handed to the clerk by one of the jurors:

"There was the lady on the NDS side, second row from the back, that I find very distracting because it looks like she's recording the trial."

I don't know if that's true or not. I'm going to not inquire any further. I don't see the lady, second row from the back, at the present time.

There's no recording of the trial. Thank you very much.

(In the presence of the jury.)

THE COURT: All right. Thank you. The jury's present. All counsel are present.

Let me state that I received a note from the jury saying that one of the jurors was concerned about a lady in the second row on the NDS side -- and, of course, that doesn't mean they're on the NDS side, but the NDS side of the court -- second from the back row, that's distracting because it appears that she is recording the trial.

Let me remind all parties that there's to be no recording of this trial. Thank you very much.

Counsel, if you would like to continue with your direct examination.

MR. HAGAN: Thank you, Your Honor. Chad Hagan on behalf of the EchoStar and NagraStar plaintiffs.

AVI RUBIN, PLAINTIFF'S WITNESS, PREVIOUSLY SWORN

RESUMED THE STAND

DIRECT EXAMINATION (Continued)

BY MR. HAGAN:

Q. Good afternoon, Dr. Rubin. Before we broke for lunch, you identified for the ladies and gentlemen of the jury the four key components of the hack methodology for EchoStar's system that the defendants developed. And you discussed briefly with them and gave us a demonstration about the I/O buffer overflow vulnerability.

I want to change focus now and talk about the three other key components. And the second one you identified was the RAM ghost effect, or address alias. Can you explain to the ladies and gentlemen of the jury what you're referring to there?

A. Sure. And I already did this to some degree before. It's basically the notion that memory is finite. Right. So it only goes so far. So what happens if you give an address that's beyond the bounds of memory?

And in the Smart Card it's a little different from a computer because you have so little memory. So when you only have a little bit of memory and if you try to address something, like you asked for a location, try to store something at a location, or read something from a location that's larger than the amount of memory you have, this particular chip had an unusual property, which is this wrapping around, of ignoring the most significant digit there.

Q. And if the RAM ghosting effect was not inherent in the chip used by EchoStar, would the I/O buffer overflow vulnerability have made a hack possible in the way the defendants developed it?

A. This particular attack relied on the RAM ghost effect, on this wrapping around. As I'll show you when I walk you through the attack, if that property had not been there,

this attack would not have worked.
Q. Let's focus briefly on the third key component, and that is sophisticated knowledge and usage of the index variable. Can you explain for us what you mean by that.
A. Sure. So again, there was a particular byte in memory -- and I'll show it to you on a chart where that was located approximately -- that had a value. And that value was important in determining when messages were read in on the communication, where they were stored in the communication buffer.

So to back up a little bit, when -- you've got your set-top box, right? That's the thing that you put on top of your TV. And it has a Smart Card in it. The way that the system functions is through messages. The set-top box, which is called the IRD, will send a message, just bits, to the Smart Card, which will then process them.

And the first thing that happens when a message comes in from the set-top box is that the bits of that message are copied into a buffer, into a section of memory that is reserved for that purpose.

How do they get there? Well, that's through this index available. The index available is incremented. Every time a byte is copied from the IRD into the memory buffer, that value increases by one. And the reason for that is so that the next byte will come right after the last byte. So this index available in memory keeps being incremented, which causes the next byte to be written consecutively.

Now, this attack relies on maliciously and intentionally changing that index variable.

So what you're doing is you're going to overflow the buffer, you're going to wrap around. And you're going to craft the message so that the index variable is not what it's supposed to be. What is it? Well, it's going to be controlled so that the next byte read in from the set-top box will go wherever the attacker wants it to go. And I'll show you how that plays into the attack.

If you didn't understand the working of that index variable, you could not have written this attack.

Q. Then, finally, the fourth key component that you identified through analyzing the defendant's hack methodology was the behavior of the exception handler. And I think you mentioned that briefly that this morning and how that related to an invalid checksum error. Can you walk us through that really quickly?

A. Sure. And I'm trying not to repeat the part I already told you.

So when the set-top box sends a message to the Smart Card in a legitimate operation, the message is computed as it's supposed to be. There are all kinds of messages sent for all kinds of reasons. One of them might be, you know, I changed the channel. I want to start decrypting that channel now. And so that causes messages between the set-top box and the receiver.

The designers of the system will program a whole bunch of different messages for operation. This is just normal use.

Now, they're worried, of course, as I said before, that somebody might have not intentionally maybe but through a glitch or communication error caused the message to be received incorrectly. And it's not that uncommon for there to be problems in hardware or software. So as a security blanket, they put in this checksum. The checksum ensures that when you receive the message, you have this way of knowing that it actually arrived intact.

If it doesn't arrive intact, there are different options. One of the things you can do is send a message back saying I didn't get it intact. Please retransmit it.

Another thing that you could do is just do nothing. Right. So if you ever push your remote and nothing happens, well, maybe that's what happens and you'll push it again. So there are different things you might do in response to the checksum being bad.

Because of that, they couldn't hard-program in the exception handling. Because sometimes you do one thing, sometimes you do another thing. So what they allowed for is flexibility in the code.

And so the way the card designers did it was they said let's just go to a particular address on the top of the stack, and we'll see what instructions that points to and then we'll just do whatever it says there.

And again, this is something like the other property where if you didn't understand that the card did that when there was a bad checksum and then -- then, you know, the attack just doesn't work. It's completely based on these four properties.

Q. Thank you, Dr. Rubin. And I know we'll get to the particular attack and the Nipper attack in a little bit.

But for right now I think it's important to talk a little bit about reverse engineering.

A. Sure.

Q. You have significant experience in reverse engineering products; is that correct?

A. That's right.

Q. Both as part of your company, Independent Security Evaluators, as well as through your work for Johns Hopkins University; is that correct?

A. That's right.

Q. And can you tell us about one of those reverse-engineering processes?

A. Sure. So there have been several of these that we've done over the years. Let me talk about the Exxon Mobil speed pass.

If you're familiar with the Exxon Mobil speed pass, it's a product which is a little device that you can put on your key chain. And you get this from Exxon Mobil so that you can go to a gas station and just hold it up to the gas pump. And you don't have to run your credit card or anything, and it will perform a cryptographic exchange.

And that's important because you don't want to be able to get someone else's credit card number used. So it's got some keys on it. And that happens really quickly. It literally takes an eighth of a second for the message to be transmitted and transmitted back.

So you hold the thing up. It's a convenience feature. And you buy your gas. And on the back end system, it looks up your credit card number, it performs some encryptions; and if you're the authorized person, then you get to buy the gas.

Interestingly, we discovered that in many Ford vehicles, like a hundred thousand of them, they were using a chip in the key that's exactly the same chip in the Exxon Mobil speed pass. And so we set out in my lab at Johns Hopkins -- and I think it basically happened over lunch one day when somebody said, "I have one of these. I wonder how secure it is" -- to make that determination.

And so we got some equipment, we set up a lab, and we basically tried hard to break this thing.

One of the justifications for doing this -- because whenever we set out to do one of these projects, which can often become controversial -- is to figure out if there will be enough benefit to society and the people for us doing this, or a net loss. And we have had projects where we start out all excited and then decide not to do it.

So I called the general counsel of the university and said we are interested in studying the Exxon Mobil speed pass. We plan on breaking it. If we're successful breaking it, we will call Texas Instruments and call Exxon Mobil.

And the reason Texas Instruments was because they're the manufacturers of the chip. They told me it would take a while, they would think about it. And then they got back to me and said, "We'll support you in that effort."

We then thought about, well, what if they're not very happy with us if we're successful, and they sue us?

So I called up the lawyers at the Electronic Frontier Foundation, which is a nonprofit organization funded, I think, to the tune of about $10 million. And their purpose is to answer these kind of questions for researchers. And they take the cases pro bono.

So I retained their lawyers, and they did a study and wrote a brief about whether or not it was ethical and whether or not it was legal to do what we were interested in doing. I had worked with them before on the electronic voting machines, and so they were familiar with my lab and how we worked, as was the Johns Hopkins general counsel.

So we received the brief back, and the brief was -- was encouraging: We wouldn't be breaking any laws.

In fact, they agreed with us that there was enough benefit to consumers when they make a decision whether or not to use this product, to understand how secure it is.

And so we set to work. And we -- it took us three months of my students working around the clock. Actually, they get pretty excited about these projects. We were sending test messages to the thing. It's basically what's called an RFID chip, a radio frequency identifier.

And if you follow the technical press, these are really up-and-coming technologies. And we -- it took us a while to be able to understand the communication. But eventually we were able to get the devices to respond to us.

And we built what we called a cracker, which is a parallel array, meaning a whole bunch of these -- and this is a very technical term, FPGA, field-programmable gate arrays. What these are is a fancy way of saying little bitty computers. And we got these little bitty computers and lined them up 16 in a row. And their purpose was to search for cryptographic keys.

And because of the nature of FPGA's, they can do it very, very fast. And so we set these things to work. And once we had understood the communication between our equipment and the speed pass, all we really needed was a big breakthrough on our cracker.

So we set up little lights on each of the crackers, and we put a webcam on it. And the thing was, once we found a key, a light would turn on, we'd see it on the webcam. So about five of us involved in the project, we're constantly looking at our computers all over the country, constantly looking at our web browsers. I was -- I remember being at a conference and sitting there with my laptop -- so I remember being at a conference and looking on my laptop all the time at the webcam to see if that light went on.

It happened one day at about 2:00 in the morning, and I got a call from one of my grad students, who woke me. He said the light went on. I waited till the next morning and went into the lab. And we had obtained the key that was in that Exxon Mobil speed pass.

Without going into a lot of the technical details, once we had that breakthrough, it gave us a lot of information, and so we were able to then reproduce that a lot faster. And we got to the point where we could crack a key in about 20 minutes.

So we felt, you know, pretty proud of ourselves. We wrote up our results. And I called the lawyers again and I said, "Well, we did it."

And they asked me what my plan was. And I had been involved in these kinds of situations before where we had something like this, and now what do we do with it.

So I called up Exxon Mobil and I said, you know, "I need to talk to somebody because we've cracked the Exxon Mobil speed pass. I'm a professor at Johns Hopkins."

And my call was returned pretty quickly. And they asked me if they could see it. And so we invited them out to the lab. We hadn't told anybody about this except the Hopkins lawyers. And they also brought Texas Instruments with them. And they showed up with a little bag full of chips. Some were Exxon Mobil speed pass, some were car key chips. We had already tested our attack on the car key chips made by Ford.

Okay. So they showed up with a bag of chips -- not a bag of chips but a bag of computer chips. And they challenged us and they said, "We're gonna have this meeting with you. We want you to tell us what you did and how you did it. But meanwhile we want you to crack these chips. We really want to see if you can do it."

So one of my grad students took it into the lab and put the cracker to work and was able to break them a lot faster for a reason that I think you guys can now appreciate, which is that they had manually entered some keys into them, but they hadn't bothered to enter any of the hex letters A, B, C, D, E, F. It was all numbers. They entered them in decimal, but the computers considered them to be in hex. And so a lot of values were impossible. And we found the keys very, very quickly.

So while we're having the meeting where we're walking them through how we broke the system, they -- you know, my student came into the room with a piece of paper with all the keys written on them. And they pulled out a piece of paper from their pocket and compared them and said, "Yes, you guys did it. Now, what?"

We said, "This is our objective. We're going to release this information, but we're going to do it responsibly. We believe in responsible disclosure." And I outlined our plan which we had come up with in advance for them. Which was, first, we were going to write a technical paper. The technical paper would describe the new science we had invented in order to perform this attack because it was things that no one had ever done before. That's what grad students are supposed to be doing. And we were going to submit it to a technical conference.

But we were going to omit the specific details of the attack. We were not going to write something that would enable somebody else to go and do it. We're simply going to show the steps that we carried out.

The next step that we were going to do is -- we knew that this would be very interesting for the media. And we didn't want to do this in an ad hoc way. And I had a relationship with a reporter at the New York Times who I had worked with on a lot of the electronic voting stories. We were going to give him an exclusive, invite him down to our lab, show him what we did, and then he would write a story in the New York Times. And then whatever happened happened. But we would not release our paper until the conference six months later.

Exxon Mobil wasn't very happy about this. They said, "Well, we'd really like it if you don't release this."

We said, "We know that you would be, but we're going to. What we want is for you guys to fix this before we released it. And here" -- and we had worked on this. "Here's how to fix it. It's a very simple fix. Use a longer key."

So that's how it transpired. The New York Times story ran. The media really picked up on this. The show 20/20 came to our lab and filmed an interview with me. And then they had a reporter walk into the lab with an Exxon Mobil speed pass in their pocket that they brought with them, and asked us to drive them to a gas station and buy gas and have that reporter pay for it. We knew that was the setup.

My students had the scanning equipment, and we asked the person to walk over here. We scanned them, got the information we needed from their FID chip. We then went into our cracker, we broke the key. We had a simulator we had built on a laptop. And we drove to a gas station with the reporter in the back seat. And we held up an antenna that we had connected to the laptop up to the gas pump. And sure enough, the tiger lit up. And we bought gas and we left.

In the story that ran on 20/20, they showed --

THE COURT: Did you get to pay for it?

THE WITNESS: No, that reporter paid for it.

At this time gas prices were getting a lot higher, and so we were inviting a lot of reporters in to do this.

In the story that ran on 20/20, the reporter showed their bill, that they had actually been the one who had paid for the gas.

So the point of this is that my specialty is really breaking systems like this. And over the years I've developed a methodology that I call responsible disclosure, of how you go about deciding whether or not to take on a particular project and then undertaking the project, informing all the parties responsible, making sure a fix can get out there, and then publishing.

And I should mention that our technical paper won the Best Paper Award at the USENIX Security Conference. I know the term USENIX has come up in this trial before. It's the premier systems organization.

BY MR. HAGAN:

Q. Thank you, Dr. Rubin.

If I understood the process correctly, you and your team at Johns Hopkins set out to determine whether or not you could reverse-engineer and crack this Exxon Mobil speed pass.

Before you did that, you thought there may be implications, so you contacted the general counsel and you contacted lawyers for the electronic -- what was it, EFF?

A. Yes, Electronic Frontiers Foundation.

Q. And who's the EFF?

A. It's a nonprofit organization which was created by donations of some very wealthy individuals to provide legal advice and support to technologists.

Q. And you notified them of the efforts that you and your team at the university intended to engage in, and they gave you a written report saying that it was okay to go down that path; is that correct?

A. That's right.

Q. And then after you determined that there was a vulnerability in this speed pass, you contacted the

manufacturer of the chip.
A. Right.
Q. And you contacted Exxon Mobil.
A. Right.
Q. And then you set up a meeting where they could come into your lab, and you could show them what the vulnerabilities were, correct?
A. Right.
Q. And then you went a step further. You actually developed a patch. You and your team developed a software patch or a way to fix that vulnerability; is that correct?
A. Right. Well, in this case it was simply advice. In other cases we've developed patches.
Q. Did you charge Exxon Mobil anything for that advice?
A. No.
Q. Did they take that advice and try to improve the robustness of their product, or did they take steps to try to improve the robustness of their product?
A. They told us that they were.
Q. If I understand your testimony at your previous deposition, you and your team at ISE also engaged in efforts to reverse-engineer the Apple iPhone.
A. That's right.
Q. And did you follow similar procedures with respect to the Apple iPhone project?

A. Yes, we followed the exact same procedures.
Q. And that was contacting the EFF attorneys?
A. We started with our own general counsel, in this case my wife. We told her what it was we were planning on doing. And she did some research. And then we talked to the same EFF lawyers that we had talked with during Exxon Mobil speed pass.
Q. And were you and your team at ISE able to effectively reverse-engineer the Apple iPhone?
A. Yes. We were able to crack it in a week.
Q. Did you determine whether or not there were vulnerabilities in that technology?
A. Yes, we did.
Q. After you made that determination -- can you tell the ladies and gentlemen of the jury the steps that you took?
A. Yeah.
So once we had figured out how to break the security on the iPhone -- and I should point out that this was a few weeks after the iPhone came out. We were planning on, you know, looking at it -- everyone. There was a lot of buzz in the media about the iPhone. And we thought, let's see what they did with security and how good it is. And once we saw that it wasn't really that good and that we could break it, we contacted Apple.
They have a web page that's devoted to people informing

them of vulnerabilities in their products. And that should give you an idea that this is not something that happens with infrequency. So we submitted a vulnerability report to them, telling them exactly where in their system the problem was and providing them with a patch that would fix it and telling them what our plans were for publication and for releasing the information, pretty much the same as we did with Exxon Mobil.

Q. And did you follow those same steps that you called responsible disclosure with respect to the Apple iPhone project?

A. Yes. I recall a meeting that I called my staff in to discuss how we were going to release the information and what our steps were going to be.

In my mind it's a pretty formal process, when you engage in this kind of research, to have a laid-out game plan for what you're going to do with respect to the release before you even really start to break something.

Q. And were you able to develop a software patch or fix for the Apple iPhone?

A. We did.

Q. And did you notify Apple of that and provide them with that information?

A. We did. Like I said, we actually sent them the patch.

Q. And did Apple eventually end up improving their product

and improving the robustness of the Apple iPhone?
A. Yes. Apple released their own patch before the date that we said we were going to publish. And they didn't use our code, but they did thank the engineers from my company in the patch, basically credited us with being the ones who found it.
I should point out that this is a relationship that large companies, software companies often have with researchers who find vulnerabilities: You show us the vulnerabilities, and in exchange, when we fix it, we'll credit you. It's kind of an unwritten rule.
Q. Dr. Rubin, are you also familiar with the term "ethical reverse engineering"?
A. Yes.
Q. Can you describe for us what that term means?
A. The way that term is used is basically to contrast the kind of reverse engineering that we do where -- I'm not talking about the iPhone now; I'm talking about for our customers where somebody has asked you to reverse-engineer something for them and has hired you to do that versus nonethical, which would be where you are doing it with some malicious intent, as a malicious hacker.
Q. Now, do you always practice ethical reverse engineering and responsible disclosure protocols in the work that you engage in with your team at Johns Hopkins and with ISE?

A. That's a fundamental goal of ours.
Q. Why is it important to practice those protocols?
A. You know, it's a very, I think, controversial and loaded activity that we partake in. And if you're not careful to first talk to lawyers and to have the studies done, you may find yourself in a precarious position where you may have done something that could get you into hot water. So we err on the side of caution.

I can give you an example. I will give you an example now of a project that my guys thought would be really cool to undertake and that we explored undertaking and that the lawyers told us not to do and that we didn't do.

You're familiar with iTunes. And iTunes has a movie rental capability now. And wouldn't it be cool to be able to kind of break or hack the iTune system so that when you rent a movie, instead of only getting it for 30 days, like is their policy, you could get it for an unlimited amount of time? That was something that occurred to one of my guys in my company. We're always thinking like that: How can we defeat this or that.

I asked the lawyers about that.

They said, "Well, there are several reasons why you shouldn't do this."

Number one, they said it was a violation of a law called the Digital Millennium Copyright Act because the content -- DMCA, for short. The content in these movie rentals is copyrighted. And reverse-engineering a copyrighted system is much more serious in the U.S. legal system than reverse-engineering something that doesn't protect copyrighted material. And this has to do with laws such as the DMCA, which were specifically designed for copyrighted material.

And so I told the guys this project's not a go and we're not going to do this. And we didn't do that one.

Q. Now, have you ever had any of your clients on behalf of ISE come to you and ask if you will engage in efforts to reverse-engineer and hack their competitors' technology?

A. Yes, we have had that happen.

Q. And have you undertaken those projects?

A. No.

Q. Why is that?

A. We're not comfortable doing that. The first name of my company is "Independent," Independent Security Evaluators. And we feel that being hired to hack someone's competitor kind of aligns us with them a little bit. And we didn't feel that we would truly be independent if we did that.

Q. Now, you were in the courtroom when David Mordinson, the defendant's engineer, one of their employees that worked on the EchoStar hack, testified; is that correct, sir?

A. Yes, I was.

Q. And you heard Mr. Mordinson testify that he didn't tell or NDS didn't tell anyone at EchoStar or NagraStar or NagraVision or Kudelski that they were engaging in the efforts to hack EchoStar's security system; is that correct?
A. That's right.
Q. And you were also here when he testified that the defendants didn't even notify the chip manufacturer that they were engaging in these efforts to reverse-engineer and hack EchoStar's security system; is that correct?
A. That's right.
Q. Did you have an opportunity to read in detail Mr. Mordinson's report, the Headend Report?
A. Yesterday.
Q. Did you find any actions in that report that discussed how the defendants could improve the robustness of their product or their security system by hacking EchoStar's technology?
A. No, that wasn't in there. It was really a description of how to hack it and an actual attack.
Q. Do you have an opinion as to whether or not the defendants followed protocol for ethical reverse engineering and responsible disclosure when they hacked EchoStar's security system?
MR. STONE: Objection. Assumes facts not in evidence, and outside the scope.

THE COURT: There's a caveat to this.

I'm going to allow his opinion, but remember, this is his protocol. Whether there's an industry-wide protocol or whether this turns out to be a violation of the law when I give you those instructions may be a far different question than the limited opinion I'm going to let him voice at this time.

Overruled.

You can answer that.

THE WITNESS: I don't believe that they followed ethical reverse engineering or responsible disclosure in that.

BY MR. HAGAN:

Q. Thank you Dr. Rubin.

I want to turn your focus now to Exhibit 998.

MR. HAGAN: Christine, if you could give him a copy of that.

And, Clint, if you could publish the first page.

(Document displayed.)

MR. HAGAN: Your Honor, this has already been admitted into evidence.

BY MR. HAGAN:

Q. Dr. Rubin, can you identify this document for the ladies and gentlemen of the jury?

A. Yes. This is the Nipper post from December 21st that was reposted on December 23rd.

Q. And is this the hack methodology that you referred to earlier in your testimony as the recipe?

A. This does contain the recipe, yes.

Q. And as part of your work in this case, did you conduct an analysis of this recipe?

A. I did. I dissected this and looked at it very, very carefully.

Q. And based on your analysis, were you able to determine whether or not there were any similar fundamental components of the Nipper hack methodology that was posted on Mr. Menard's website and the hack methodology developed by the defendants?

A. Yes, I was.

Q. And can you identify those for us?

A. Sure.

So when I looked through this code and basically through this hack, it was really kind of deja vu because it has the exact same four components that I listed earlier that are in the Haifa attack.

It exploits a buffer overflow in the exact same way that the Haifa report does. It relies on this unknown property of the ghosting effect to wrap around memory to get to the index variable, relies on the index variable and demonstrates a complete understanding of the operation of the index variable. And also has an incorrect checksum at the end of it which causes a jump to the address that's on the end of the stack, which happens to be the right place in the attacker's code.

So as far as I could tell, there was a blueprint that both of those programs were following. And the thought that came to my mind was that it's the same DNA. These are a strain of the same thing.

Now, the code in the two programs is different. You know, it's really the structure that's the same. These -- the methodology, these four things are the same. And so in my mind it was as though the program was written twice maybe by different people or maybe by someone who had learned more about the chip when they wrote the second one.

Q. Now, as part of your work, did you compare this hack methodology, the Nipper hack methodology, with the defendant's hack methodology?

A. Yes, I did.

Q. And can you describe in terms that we can understand and get our arms around -- can you describe how you went about that analysis and comparison?

A. Sure. At this point it might help for me to use the demonstrative.

MR. HAGAN: Your Honor, may I publish the demonstrative?

(Document displayed.)

THE WITNESS: Okay. What I have here, this is the code in Mordinson's report, the Haifa attack. As I told you earlier, it has the 0X and then hex values throughout.

And over here is the code from the Nipper posting. We've looked at that posting several times. And this is what was in there as how to attack the card.

What I want to do is walk you through. On this chart I have memory. And I have two identical copies here. And this would be the RAM of the Smart Card. And what I'd like -- the purpose that I have in showing you this is to demonstrate this DNA property, the similarity of these two attacks. And it's technical, and I'm going to go slowly.

So the IRD sends this message to the Smart Card. Right. Just to show you -- so when it sends hex 21, right, that's 0010, 0001. It's our shorthand. Right. And so each one of these corresponds to a whole bunch of zeros and ones, and we're not gonna say that again.

So what happens is, in RAM there's this one byte here which is just hanging in the middle there. And this -- I haven't seen something like this before. It's unusual. And this is the index variable. And it's got a value.

And what happens -- now I'm going to be the computer, okay. I'm the computer inside the Smart Card, and I'm going to do what the computer inside the Smart Card is gonna do. I'm the Smart Card and I received this message and I'm going to take the first byte that I'm receiving and I'm going to consult the index variable, say, "What's the value in the index variable?" "Thank you."

And now I'm going to use a formula that I have in my code to calculate, using the index variable, where should I put this byte that I just read.

And it comes right here. 21. In fact, this last box here is what's called the I/O buffer. Input/output buffer. This is where the Smart Card places things that it reads from messages.

And the next thing I'm going to do after I read in this byte is I'm going to increment the index variable. I'm going to add one to it. Why? So that I'll copy the next byte there. Let me do that. Don't worry, I'm not going to do all of them.

Zero, zero. And on and on.

And so what I'm going to do is I'm going to copy all of these bytes into here. But guess what? There's a problem.

The problem is that when I get to this byte right here, this zero, zero -- and I'm the one that put these lines here. This is just all the numbers.

When I get to here, I'm now here. I'm at the end of the buffer. Now, someone sending a message this long is intentionally overflowing the buffer. Right. This is gonna overflow the buffer.

And so what's gonna happen to this byte? Well, the developers of this -- of this card assumed that when something got copied past the end of the buffer, it would be written off here. It would fall out of memory, right. Because they didn't know -- and we heard testimony yesterday that they didn't know about the RAM ghost effect.

So, in fact, these bytes don't get copied over to here because of the RAM ghost effect. They get copied over to here.

Now, I'm going to tell you something about this card. The first hex 20 bytes, which is 32 in people talk, the card does not actually let you write to the memory. It's protected. So what happens when these zero bytes are written to here? Nothing. Nothing changes in the memory. It's just an operation that the attacker had to perform to get to the index variable. This is like a race to the index variable. The attacker's goal is to change that index variable to make the attack work.

So I've marked off this section of zeros. The attacker knew that these wouldn't actually have any impact whatsoever, so they just put zeros there.

And when I get to this point, I'm now at the end. I just didn't write zero, zero there. And those bits go off into thin air.

Now, once I'm done with that, this area of memory is no longer protected. So now we're actually gonna copy. And remember, we keep incrementing the value in the index variable.

So this zero, zero will show up here, and these values will be consecutively copied into memory up until this point.

And notice that they're not all zeros. There's 81, there's a 5, right? So in here somewhere you're getting 81, you're getting 5.

The attacker was very clever, and he had all the code to the chip, so he knew how it worked and realized that there were actually valuable things in memory over there. You can't just go around overwriting memory. Okay.

So in order to preserve the functionality of the card so the hack would work, the attacker, David Mordinson in this case, had to put legal values there that the card would be able to operate with.

Now, as an attacker this can be done through trial and error. Or you can sit down with a microscope and tweezers and look at the code, figuratively. And look at the code and figure out that needs to be an 81. And I'm not sure how he did it. Maybe some combination of those.

But anyway, it turns out that these values that were in the Haifa report are the correct values for this attack to work.

The next thing is DF. And this is where it gets clever, and this is the trick. DF gets written into the index variable.

What that does is -- this was figured out using the complex formula. With basic algebra you change the unknown, right. We're trying to get something to a particular spot here. And we know that DF is going to come in and get us there. Right. So that's basic algebra.

And where the attacker wanted to be was right here. Let me see which I drew as the stack. This is the stack.

Now, for the purposes of this case, it's not important for you to know what a stack is, and I'm not going to tell you. But it's a section of memory used for things.

And what the attacker wanted is for 019C to be at the top of the stack. And so he crafted DF so that the next byte that could be copied, which was here, would get you right there. Okay.

Now, why did he put it in twice? I wondered about that. I looked at it a lot, and I said he wasn't sure about this value, so he wanted to make sure, if he overshot, he'd still get it. That's my theory. Anyway, it doesn't matter. You put it in twice, belt and suspenders, you're gonna get there. In fact, it does work.

Now, what is 019C? Well, interesting, that's the address of the I/O buffer. And that is where all of this code was copied.

So I actually misspoke. This first line here is the header, and it doesn't get copied. So we start copying here at 9D, and we copy all the way to 00. That's this.

Okay. So you're getting a feeling, I hope, for how intricate this attack is, how much it relies on the index variable, the ghost effect, the buffer overflow.

There's a signature here to this attack. And it gets you over here to 019C. And then, guess what, we hit a bad checksum. I'm the computer again. I say, well, gosh, the checksum for this message is not FF. What am I supposed to do? Well, I'm programmed to look at the top of the stack, grab that address and start executing code there. Right here. That code came from right here.

And so what is this? Well, if you look at the Haifa report, the Headend Report, this is code that dumps the EEPROM, okay. This is software code that could have been written on the card originally. We call it shell code.

And this should also help you understand that this is a generic attack. Once this is known, an attacker can put whatever program code they want here, and it will run on that card because of this attack.

Okay. If you have trouble following that, you're gonna see it again now because the Nipper attack is very, very similar to the Haifa attack.

It is also different in some ways. And the differences, to me, demonstrate that it was either somebody else who looked at this attack and said, "I'm gonna do the say thing," or the same person learning a little bit more about this chip took some shortcuts. And you'll see the shell code is shorter. It's actually better written, it's more compact, but it does the same thing.

Okay. So I'm going to shift gears now, and I'm gonna talk about this posting which was on this website on the Internet.

So let's start off. We have the same header. And we start copying bytes. Now, what's kind of interesting is notice that these bytes don't look like code. 1, 2, 3, 4, 5, 6, 7 -- it's wasted.

Okay. There is no objective to these bytes except to get you to the index variable. Okay. Clearly the programmer knew about that.

So what I've drawn here is fill the I/O buffer basically with garbage, and you get to the 04 at the end. And so far the attacker has done nothing except get prepared to overflow the buffer.

All right. Well, now the attacker's gonna overflow the buffer, and the attacker knows that it's gonna wrap around and start writing things here. Only the attacker also knows that these values don't matter. They actually won't be written to the card because the first 20 hex bytes don't write. The card's protected. So you basically can see zero, 1, 2, 3, 4, 5. The attacker doesn't really care.

So we've gotten all the way down to here in this message, and all we've done is try to find that index variable.

Now, once we get to here, we start writing into memory here. So we're going to write 0001, et cetera. None of these values really matter either. Except some of them do. Some of them have to overwrite legitimate values just like Haifa did. Right. And so he didn't use 81 and 5 because there are other values that will not crash the card either that are legal values.

And so those other values are written here. And then the index variable is written.

So at this point the attacker's actually done something really important.

Now, just like DF, actually where the attacker wanted to be, C3 is gonna get you to where the attacker wants to be. And where does the attacker want to be? This attacker decided -- this code decided to be on the stack.

And again, it doesn't matter what the stack is. But C3 was done very, very cleverly so that the next byte would be written in the middle of the stack somewhere in a very precise location.

Just to remind you that the index variable tells the computer on the chip where to write the next byte.

Okay. So once we overwrite this, we no longer keep writing over here. We're jumping over to here. And we write the 9B, 9C. And this was actual code, so it's gonna have more structure to it. It's not 1, 2, 3, 4, 5.

And this code was written so that right when you get to the end of it, you're going to write 0060, and that's gonna show up at the end of the stack.

Okay. So the attacker counted backwards to get the shell code written to a place where once it was all copied, the last byte would be at the top of the stack.

And then an incorrect checksum. The checksum for this message is not 55. So what happens when an incorrect checksum is reached? When an incorrect checksum is reached, we're gonna go to the top of the stack and start executing at that address. Well, what does that address point to? Right there.

And so it's gonna start executing this code.

Now, let me point out why I believe that these are pretty much the same program and why they're so similar.

Both of the programs require you to overflow a buffer. Both of the programs rely on the ghosting effect which wraps you back around to the top, which was a property of this card that was not known.

Both of these attacks target the index variable. They race to the index variable so that they can put the precise value that they need to, to get exactly where they need to be in the code. And both of them orchestrate an address at the top of the stack, and they give an incorrect checksum so that you will go to that address and start executing the shell code.

Okay. I'm not a biologist, but this is the same DNA to me. These two things, to me. This could not have been done without access to the Mordinson report.

BY MR. HAGAN:

Q. Thank you, Dr. Rubin.

Now, Dr. Rubin, you were also present when Mr. Mordinson testified, and we created a little spreadsheet, a little matrix; is that correct, sir?

A. Yes.

Q. And in the matrix or the table that Mr. Mordinson and I created, we identified several similarities in the Haifa and Nipper hack. The ones that Mr. Mordinson identified were the reference to Nipper, the ATR software application that the defendants developed for this project, the application of this to a ROM 3 EchoStar card.

Those are not important to your DNA analysis; is that correct?

A. That's right.

Q. The bottom four are the ones that Mr. Mordinson identified. Dumping the EEPROM, overflowing the buffer, using the RAM ghost effect and executing code in RAM.

Mr. Mordinson did not talk about the index variable.

Did you find any reference in the defendant's Headend Report that talked about the index variable and how that worked with EchoStar's security system?

A. Yes, I did. He actually in his report identifies that particular byte as the index variable and explains how it's used to form the attack. So it's an integral, necessary component in his report.

Q. Mr. Mordinson also testified that both of these attacks use the ability to execute code in the RAM portion of the card. Is that consistent with the demonstration that you just gave us?

A. Right. Remember that the address that's obtained at the top of the stack jumps to some code which is in RAM and starts executing it.

Q. Now, you've had an opportunity to review the report submitted by the defendant's software expert in this case, Nigel Jones; is that correct?

A. That's right.
Q. And in his report Mr. Jones identifies -- he does the opposite as you. He tries to identify a lot of technical differences between the two hack methodologies. Did you review that portion of the report?
A. I did. I read his whole report.
Q. And can you explain to the ladies and gentlemen of the jury why those technical differences are not significant in your opinion and in your analysis?
A. Sure. So let me give you a few of the things that he points out. He has a page where he talks about that the code in the Nipper attack and in the Haifa attack is of different lengths, which, if you looked at these, you would notice that.

The reason for that is that the Nipper one puts in a whole bunch of nonimportant data before it gets to where it needs to be; whereas, the Haifa one puts the shell code right in the I/O buffer. So they're obviously not the same program. They do it a little bit of a different way. So they're not going to be the same length.

Another thing that he points out is that in the shell code itself, the Haifa methodology writes actual assembly language code to do the writing of the data from the -- when it's writing the EEPROM.

And the Nipper methodology uses a built-in function of the card, so the code is more compact and more efficient. And so he uses that to try to make the case that these programs are different.

In my opinion, if you are developing a program like the one in the Headend report, which is clearly for demonstration purposes because of the detail in which the report explains things, you might do things more pedagogically.

And when you're building an attack that you expect to work in the real world, now you've had two years since the Haifa report was written to improve on it. So that's why I think it's either the same programmer learning more and writing a better attack or someone else looking at the Haifa report and then building their own from scratch.

Q. But besides these technical differences, the fundamental components of each of the hack methodologies is materially identical; is that correct?

A. That's right.

Q. And I understand from Mr. Jones' report that there are different code lengths and different styles of coding. If I understand your testimony correctly, that's no different than if two people wrote the same sentence with the same words but some inverted nouns and verbs and some used different types of penmanship; is that correct?

A. That's right. It's not inconsistent with two different people writing the program from the same source to have different programming styles.

Q. Now, Dr. Rubin, after comparing the defendant's hack methodology for EchoStar's security system and then the Nipper hack methodology posted in December of 2000, is there any doubt in your mind, based on all of your years of experience and analysis in this case, that the Nipper hack methodology was derived from the defendant's Headend Report?

A. I don't have any doubt. In fact, if two students in one of my classes were to turn in these two messages as their assignments, I would give them both an F because there's no way that that wasn't collaboration or copying.

MR. HAGAN: Thank you, Dr. Rubin.

Pass the witness, Your Honor.

THE COURT: This would be cross-examination by Mr. Stone on behalf of NDS.

MR. STONE: Thank you, Your Honor.

CROSS-EXAMINATION

BY MR. STONE:

Q. Good afternoon, Dr. Rubin.

A. Good afternoon.

Q. We've met before a couple times, right?

A. Right.

Q. Now, you sat through the entire trial so far, correct?

A. Pretty much. I missed a few minutes here and there.

Q. You were here when Mr. Nicolas testified, correct?
A. Yes.
Q. And you heard the suggestion that because someone posted using the name Nipper in late 1998 -- I think it was December of '98 -- that that was somehow proof that Nipper could only have come from the Headend Report which preceded that. Do you recall that?
A. I don't remember him saying that that was proof of that, but I do remember the discussion.
Q. And you know from your own report that such a suggestion would be incorrect, right?
A. What's the suggestion?
Q. That Nipper could only have followed the Headend Report and no other source because the Headend Report preceded it.
A. Okay.
Q. Would you like to see the report?
A. No. I just want to understand the question.
Q. Let me show you Exhibit 799, which is your report.
A. Okay.
Q. That might help. What I'm focusing on is page 799-19 for the time being.
MR. STONE: And, Your Honor, I'd like to publish just that portion of it.
THE COURT: Well, I'm not certain, Counsel. As long as this is going to be, you know, published by both

sides, different portions of different doctor's reports, and you reach that agreement, I have no concern. Normally reports don't go up.

MR. HAGAN: That was my understanding, Your Honor. I thought we had already discussed this, and each side's expert reports were not going to come in or be published.

THE COURT: I did, too.

MR. STONE: I was just going to show a portion.

Let me do this: I can focus him on a reference, and we can go from there. That will work.

BY MR. STONE:

Q. Dr. Rubin, looking at page 19 of your report, you have a reference there to the first known case of an EEPROM dump attributed to Swiss Cheese Productions on October 24th, 1998.

Do you see that?

A. Yes.

Q. And you have an NDS number, which is a Bates stamp number for a document produced in this case, correct?

A. Right.

Q. 1760?

A. Right.

Q. And that would have been a document that you relied upon to prepare your report?

A. Yes.

Q. Okay. If I could show you Exhibit 225, which is that document. If you go to page 225-02, that's the -- pardon me -- 225-03. That's the Bates stamp number you were referencing, correct?
A. Right.
Q. And this was attached to the e-mail that is at page 225-01, correct --
A. Yes.
Q. -- if you look at the document?
A. Yes.
Q. And this is a document you relied upon in reaching your opinions, correct?
A. Right. I looked at this, yes.
MR. STONE: Your Honor, I would ask to publish Exhibit 225 and move it in at this point.
THE COURT: Any objection?
MR. HAGAN: No, objection.
THE COURT: Received, and you may publish the document.
MR. STONE: Thank you, Your Honor.
(Exhibit No. 225 received in evidence.)
BY MR. STONE:
Q. Looking at the page that is 225-01, this was an e-mail from Ted Rose to Reuven Hasak and another gentleman, Yoni Shiloh, on October 25th, 1998.

Do you see that?
A. Yes, I do.
Q. In reviewing the documents in this case, did you learn that Mr. Rose was responsible for monitoring the Internet?
A. I don't remember that, but I -- sure looks like it.
Q. And it shows down at the bottom, No. 3, EchoStar CAM dump and public keys by Swiss Cheese Productions.
Do you see that?
A. I do.
Q. All right. And that was the file that was attached that you looked at for the purposes of your report, correct?
A. Yes.
Q. And if you could go to the next page, 225-02, please.
A. (Complies.)
MR. STONE: Could we get 225-02.
(Document displayed.)
MR. STONE: If we could zoom in a little on the top.
(Technician complies.)
BY MR. STONE:
Q. And this was part of the text file that was posted on the Internet, correct?
A. Right.
Q. And it says, "Dump from running CAM REV3.13." You understood that to be from a ROM 3 version card?

A. That's right.
Q. And then if you go to the next page, please, 225-003, that's the EEPROM dump that appeared on the Internet sometime before October 25th, correct?
A. That's right.
Q. And do you see a phrase in there that looks familiar to you from the code that EchoStar and Nagra use?
A. Yes, I do.
Q. And what is that phrase?
A. You're going to make me say it. All right. "Nipper is a butt licker."
Q. And who do you understand to have put that in the code?
A. The creators of the card.
Q. The creators of the card or EchoStar? Do you know?
A. Well, I heard testimony yesterday that EchoStar asked Nagra to put it in.
Q. Do you know how many employees at EchoStar had knowledge of that phrase?
A. I do not.
THE COURT: Well, which portion?
MR. STONE: The Nipper --
THE COURT: "Nipper is a butt licker"?
MR. STONE: That phrase. That pass phrase.
THE COURT: Is that correct?
MR. STONE: That's correct.

THE WITNESS: What was the question? I thought you asked me how many people knew, and I said I didn't know.

BY MR. STONE:

Q. And the Headend Report was completed on November 1st, 1998, correct?

A. That was the final draft, yes.

Q. Well, the previous draft was October 27th, 1998?

A. Right.

Q. And we know that sometime prior to October 25th, 1998, there was on the Internet by the Swiss Cheese Productions a dump of the EEPROM with the phrase, right?

A. That's correct.

Q. So anyone visiting that site prior to October 25th, 1998, would have seen that phrase in the code that was posted, right?

A. That's correct.

Q. Now, is it correct that -- it's your understanding the lawyers for plaintiffs found you as an expert in this case by doing a Google search?

A. That's my understanding.

Q. And prior to this lawsuit, had you ever evaluated the security of a conditional access system for satellite television?

A. No, I had not.

Q. And the Smart Cards in question, the ROM 3 cards, are

considered embedded chip systems?
A. That's right.
Q. And have you ever written a software program for embedded chip systems?
A. No, I have not.
Q. Or have you ever written software for any embedded chip?
A. No.
Q. And am I correct, you haven't done a code review for an embedded chip?
A. Not -- not -- no.
Q. And you've never been called upon to design a Smart Card, correct?
A. That's right.
Q. And the ROM 3 card had an ST microchip with a Motorola 6505 processor?
A. I thought it was a 6805.
Q. 6805. I'm sorry, you're right. Is that correct?
A. Yes.
Q. Have you ever written a program for an ST microchip?
A. No, I have not.
Q. Have you ever written a program for a 6805 processor?
A. No.
Q. And I believe you testified that you're not even familiar with the assembly language that's used in the chip

in the ROM 3 card, correct?
A. I wasn't before this case.
Q. In fact, the first time you ever reviewed assembly language code was part of your review of the code on March 28 -- 26th, excuse me, of 2008, correct?
A. That's not correct. I've been working on assembly code for years, and I taught a course with students working on assembly code.
Q. Was that the first time you'd ever reviewed assembly code for a Motorola 6805 processor?
A. Yes.
Q. And that has different operations codes than other processors, correct?
A. Yeah.
I should explain that every chip has its own flavor of assembly language. The hard part is understanding conceptually how to do assembly code. That's what we teach.
Once you've worked with a particular assembly, understanding a different one is the matter of the syntax. Like are you copying things from the first parameter to the second, or the other way around?
But the question is -- in IBM chips, for example, if you see loader register, for example, and then there are two values, you copy the second value to the first. And in the Motorola assembly language, you copy the first to the

second.

So when you work under an assembly language, you need to get the reference manual and you need to understand what the syntax is. But once you've done a lot of assembly language programming -- and in my case I taught assembly language programming -- looking at a new one is just a matter of looking a few things up.

Q. Now, when you did the code review, though, you spent about an hour and a half reviewing the ROM 2 code before you realized you were reviewing the wrong code, correct?

A. I don't think it was that long, but it was a bit of time where I was looking at the ROM 2.

Q. And that's because you didn't have familiarity with the code used in this particular chip, correct?

A. No. Actually the reason that I started looking at ROM 2 was that I was given a machine with all the code on a hard disk but no information about the file system structure. So I didn't know which code was where.

And once I started looking at it -- and I think I noticed a comment in one of the files -- or something triggered that I might be looking at the wrong code. And then I went back and figured out where the ROM 3 code was.

Q. Now, you've never set out to determine how cheaply or how efficiently one could extract the code from a NagraStar Smart Card, have you?

A. No.
Q. Am I correct, you did not read the deposition of Christopher Dalla in this case?
A. That's right.
Q. Do you know who Christopher Dalla is?
A. I don't.
Q. And so therefore you don't know what Mr. Dalla testified to, about when he obtained the ROM 2 and ROM 3 code, correct?
A. I have no knowledge of that.
Q. Have you ever heard of the DISH Plex piracy group?
A. Just in court.
Q. Before coming to court here this week and last week, had you ever heard of that group?
A. I think you asked me that same question in one of my depositions, but other than that...
Q. Have you ever heard of the EROM hacking forum that was hosted by the DISH Plex piracy website?
A. Same answer.
Q. Did you read any of the testimony about the DISH Plex piracy lab in Thunder Bay, Ontario that was used to extract code from EchoStar access cards?
A. No.
Q. Did you read the deposition testimony of Mr. Pilon, who's one of the plaintiff's informants, about the DISH Plex

piracy lab?
A. No.
Q. Did you review any documents that plaintiffs provided you about information they had about a piracy lab in Thunder Bay, Ontario?
A. No.
I should mention that my scope of my work was to look at the Haifa report and the Nipper posting and to draw a comparison, and not to read a lot of depositions that aren't related to that.
Q. Did you find out information about ROM dumps of the ROM 3 card that were available on the Internet prior to December 2000?
A. I found information that there were some fragments available.
Q. Did anyone provide you documents that showed you that there was an entire commented disassembly of the ROM 2 code on the Internet going back to early 2000?
A. No. Everything that I looked at in this case related to the ROM 3.
Q. And did you analyze any piracy devices before being retained in this lawsuit?
A. No.
Q. And did you conduct an investigation to specifically determine how much of the ROM code of either ROM 2 or ROM 3

were on the Internet prior to December 2000?
A. No.
Q. And you understand that defendant's experts undertook that exercise?
A. Yes.
Q. And that's not something that you attempted to do or to rebut; is that correct?
A. No, that's right. I actually relied on Michael Barr's report, B-A-R-R, which deals with that.
Q. Now, you're the founder and president of Independent Security Evaluators, correct?
A. Yes.
Q. And your firm is expert in software, reverse engineering of secure products?
A. Yeah, any software.
Q. And occasionally you've been hired to do third-party evaluations, which means breaking into someone else's system to evaluate it, right?
A. No. We have been hired before to do third party, and we only do that work with the stipulation that it's with the cooperation of the third party. So we've done it in the case where somebody wants to buy a product and they want us to make a judgment on how secure it is.
Q. Well, your company has done some reverse engineering that is unknown to the party whose product or system you

were reverse-engineering, correct?
A. I should point out that's getting into the government work that I'm not supposed to talk about.
Q. And haven't you done that also in the private sector, sir?
A. It's under circumstances that I'm, again, not supposed to talk about.
Q. Have you ever attacked a product just to see if you could do it?
A. Yes.
Q. And you've done that more than once, correct?
A. Yes.
Q. And you consider yourself a hacker, right?
A. I consider myself a White Hat hacker. There's a distinction between a malicious hacker, who is what's been called in this case a pirate, versus a tinkerer who likes to play with computers and look at where things break.
Q. Are there other companies that you're aware of that offer hardware or reverse-engineering services in the private sector?
A. Is it "or"? Hardware or reverse engineering?
Q. Hardware or software.
A. Yes.
Q. Have you heard of Semiconductor Insights?
A. No.

Q. Are you familiar with Michael Strizich, S-T-R-I-Z-I-C-H?
A. No.
Q. And have you ever or anyone at your firm ever presented at the Black Hat conference?
A. Yes. Dr. Charlie Miller, who works for me, has presented there.
Q. And have you attended Black Hat conferences?
A. I've never been.
Q. Does the term "Black Hat" refer to malicious hackers or pirates?
A. The term Black Hat sometimes is used to refer that way.
It was pointed out earlier in this case that the term Black Hat has broadened over time. The Black Hat conference is as likely to include FBI agents, for example, as it is pirates. It's a place where all kinds of people get together to discuss issues around breaking system and hacking.
Q. Well, it's a conference of folks who are involved in the security business, right?
A. Right.
Q. And didn't you testify that the Black Hat conference was the opposite of what the name implies, that it is not malicious hackers?
A. No. Malicious hackers go there. It's not intended for

malicious hackers. I don't think those kind of meetings are held in the public.
Q.   Why is the name Black Hat associated with a conference of security researchers?
A.   I think its history. I think that there used to be a bunch of people that you would be more likely to call real hackers, malicious hackers, that would get together. And I think that evolved over time into a legitimate conference.
Q.   Do you know where the name Black Hat originates with respect to computer security?
A.   I do not.
Q.   And the Black Hat conferences would be where people gather to discuss hacking techniques and reverse engineering and countermeasures, correct?
A.   Yes.
Q.   And so if somebody suggested that the term Black Hat in conjunction with hardware or software security engineers necessarily means something malicious, that would be incorrect, right?
A.   Are you talking about that specific conference?
Q.   Yes.
A.   Yes.
Q.   Now, let me show you, if we could, Exhibit 809.
MR. STONE: This is in evidence.
(Document displayed.)

BY MR. STONE:
Q. Okay. Now, you've seen this article before, I take it?
A. I believe you showed it to me at my deposition.
Q. And looking at the first page, you're familiar with Mr. Kuhn, right?
A. Yes. Marcus Kuhn is a well-known security researcher.
Q. And he works at the laboratory at the University of Cambridge, the Smart Card laboratory?
A. I'm not sure if he's still there, but he used to.
Q. And another professor at Cambridge was Ross Anderson?
A. Right.
Q. And Mr. Kuhn came up with techniques that were publicized to work on the ST micro family of microcontrollers or chips, correct?
A. I believe that's right. I'm hedging because I'm not sure if it's that particular chip or not.
Q. Well, the ST microchips are the ones used in the EchoStar access card, correct?
A. That's right.
Q. And you respect Mr. Kuhn's opinions in the area of extracting code from embedded chips, right?
A. I what?
Q. You respect --
A. I respect, yes, definitely.
Q. -- his opinion about extracting code from embedded

chips?
A. Yes.
Q. And if I could direct your attention to the first page, at the bottom, it's in small type.
A. Yes.
Q. This paper was presented at a USENIX workshop?
A. Right.
Q. And that was an organization you were a member of?
A. Yes. I still am.
Q. And were you on the board of USENIX as well?
A. Yeah. I spent four years on the board of directors.
Q. Were you on the board when this article was published?
A. I think so. I think I was on the board until 2000.
Q. And you would agree, USENIX is a legitimate organization?
A. It's the top academic organization for systems security.
Q. And were you present when this paper was presented at the USENIX proceedings in May of 1999?
A. No. USENIX has about 12 workshops every year in various different areas. And I never miss the USENIX security one, but I don't go to all the workshops.
Q. If you look at the paragraph entitled Abstract.
A. Right.
Q. It says, "We describe techniques for extracting

protected software and data from Smart Card processors. This includes manual microprobing, laser cutting, focused ion beam manipulation, glitch attacks and power analysis." Do you see that?
A. I do.
Q. And those were all well known as of May 1999, correct?
A. I would imagine they became well known after this paper was published.
Q. Did Mr. Anderson and Mr. Kuhn publish papers before this that discussed some of those similar methods?
A. I don't know.
Q. And it says, "Many of these methods have already been used to compromise widely fielded conditional access systems, and current Smart Cards offer little protection against them."

Did you notice in this article that they went on to provide countermeasures to protect Smart Cards against these types of attacks?
A. I would have to look again. I'm only familiar with this paper from my deposition.

I imagine that USENIX's paper would require you to talk about that in it, so I have no doubt that they talk about that. I just can't say without looking.
Q. Okay. Do you know whether Kudelski Nagra implemented any of the countermeasures identified in this article?

A. I don't know which countermeasures were identified in the article.
Q. Did you read any of the references at the end of the article as part of your work in this case?
A. No. I only looked at this briefly in my deposition.
Q. Now, your company hacked the Apple iPhone, correct?
A. That's right.
Q. And it was three weeks within its release?
A. Something like that.
Q. And your company was the first to hack the iPhone, correct?
A. Well, other people had been able to unlock it, which is different from what I consider hacking it. But we were the first to be able to run shell code on it.
Q. And it was your idea to hack the iPhone, right?
A. Yes.
Q. And you thought it would be good for your company if your employees were able to hack the iPhone, right?
A. Yes.
Q. And one of your purposes in hacking the iPhone was so that your company would get credit for doing it, right?
A. Yes.
Q. And that included publicity, right?
A. Right.
Q. And publicity generates business, doesn't it?

A. It does.
Q. And when your company hacked the Apple iPhone, what was the vulnerability you exploited?
A. There was a buffer overflow vulnerability.
Q. And that's one of the most common forms of attack in computer systems, correct?
A. That's right.
Q. And anybody with any expertise in computer security would know about buffer overflow attacks, correct?
A. Absolutely.
Q. In fact, a lot of them were publicized in the 1990's, correct?
A. That's right. They get publicized almost every day.
Q. Is that why your company first thought of using that kind of an attack on the iPhone, to crack it?
A. If you're going to try to break a system, I think buffer overflow is the first thing you might try.
Q. And that was, in fact, the first thing that Mr. Miller of your company tried on the iPhone, right?
A. That's right.
Q. And he was able to extract the binaries for the application code from the iPhone as part of his hack methodology, right?
A. Actually, he used other tools to do that.
Q. What tools did he use to extract the binaries?

A. I believe -- oh, I'm not supposed to say that.
I think he used the Jailbreak program.
Q. And what was Jailbreak?
A. It was a tool that ran on the IPhone that would allow you to take binaries off of it.
Q. Was it a malicious program?
A. Depends how it was used. It was a tool, and all it did was let you take binaries off.
Q. Was it an official Apple program?
A. No.
Q. Was it written by hackers?
A. I don't know.
Q. Did you use any of that work or any of that program in your work?
A. Dr. Miller did use Jailbreak, if that's what you're asking. I already said that. Do you mean me or ISE?
Q. You personally, sir.
A. No.
Q. And the binaries are maintained in the firmware of the Apple iPhone, correct?
A. Right.
Q. Could you explain what firmware is?
A. Sure. In a device firmware is a layer between the software and the hardware. So the -- if you think about an iPhone, you've got your e-mail application, your phone

application, stocks; these are applications. You've got the hardware, which is physical wires. And then what's in between is kind of -- in a computer you would call the operating system, is called firmware in a device like the iPhone.

Q. And getting the machine code out of the iPhone was the first step to disassembling it, to understanding the code, correct?

A. That's right.

Q. And that was the process that your company did to hack the iPhone?

A. Right. You have to learn something about it before you can hack it.

Q. And how was your company able to extract the machine code from the firmware of the IPhone to discover the buffer overflow vulnerabilities?

A. That's not exactly how we did it.

Q. Well, didn't you use fuzzing to extract the binaries?

A. I'm going to help you out a little bit. No. We used fuzzing in a different way. But we extracted the binaries simply using Jailbreak.

Q. And how is the buffer overflow vulnerability discovered by Mr. Miller?

A. He discovered it with fuzzing.

Q. Fuzzing is where you send invalid inputs to the

computer and you get the results back and you're able to determine whether the buffer will overflow, correct?
A. It's a -- fuzzing is a trial-and-error concept. It's a little broader than just what you described.
In general, anytime you -- like, remember that website that I showed you when I taught you what a buffer overflow was? If you were to take a web page and just start putting in all kinds of crazy random things and see if something good happens -- like you get a big raise -- then that's fuzzing.
But in order to determine whether or not there's a buffer overflow, there's a very directed way that you would fuzz. And that's what Dr. Miller did when he was hacking the iPhone.
Q. And so fuzzing can be used to discover a buffer overflow vulnerability, right?
A. Yes.
Q. And your company uses fuzzing to evaluate secure products, right?
A. It's a tool that we use.
Q. And fuzzing has been around since at least 1989, correct?
A. Yeah. Longer than that. I mean, the term fuzzing hasn't been around longer than that, but the concept has been.

Q. And in opening statement, counsel said your firm had been retained by Apple. Did Apple retain your firm to hack the iPhone?
A. No.
Q. Did you have Apple's permission to hack the iPhone?
A. No.
Q. Let me be clear, then. Before you hacked the iPhone, you did not call up Apple's general counsel and ask them if it was okay to do that; is that right?
A. That's right.
Q. And you didn't have a brief prepared before you hacked the iPhone, did you?
A. Yeah, we did.
Q. Did you contact anyone at Apple for their permission before you published any results?
A. Not for their permission. We basically told them about it before we published the result.
Q. And at the time your company hacked the iPhone, there were lots of malicious hackers trying to hack the iPhone, correct?
A. That's right.
Q. And so after you hacked the iPhone, you didn't maintain the results in some encrypted form on a secure computer; is that correct?
A. No. That's how we keep all of our information.

Q. And how long did you keep it encrypted on the computer?
A. Basically -- what do you mean by "information"? The codes that we wrote for the hack or the report that we wrote?
Q. All the information relating to how you conducted the hack and the code that you wrote to execute the hack.
A. So we never released the code except Charlie gave a few snippets of it in the talk that he gave. And the report we released I think three weeks -- I don't remember the timing exactly, but we gave Apple three weeks before we went public with this.
Q. And was Apple happy that you had hacked the iPhone when you had first told them?
A. No.
Q. Let me show you Exhibit 800, please. Okay. Do you recognize Exhibit 800?
A. Yes, I do.
Q. Is this the first report that your company published on hacking?
A. I think this was our only report about it. I don't remember another one.
Q. Well, do you recall a presentation at the Black Hat conference that Mr. Miller gave?
A. Right, yes.
MR. STONE: Your Honor, I would move Exhibit 800

at this time.

THE COURT: Any objection?

MR. HAGAN: No objection.

THE COURT: 800 is received.

(Exhibit No. 800 received in evidence.)

BY MR. STONE:

Q. All right. Now, this report is entitled "Security Evaluation of Apple iPhone," correct?

A. That's right.

Q. And that's the same thing as hack?

A. No. That's not the same thing as a hack.

Q. Did you ever use the term "hack" with respect to what you did to the iPhone in any publications?

A. I would imagine. I don't have any problem calling it that.

Q. Now, look at the bottom of the second page, please, sir -- 800, Page 2 at the bottom.

It says: "Apple was notified of these findings including detailed technical documentation on July 17th. While this paper serves to highlight our findings, we will not release the remaining technical details until August 2nd. This delay is provided in order to give Apple sufficient time to produce patches so that hackers cannot take advantage of these vulnerabilities." Right?

A. Right.

Q. You gave Apple two weeks, approximately, to get a patch together?
A. There was more than that. We actually wrote a patch and gave it to them as well.
Q. Isn't it true, Dr. Rubin, that your company was going to publish this information about hacking the iPhone whether Apple issued a patch or not?
A. That's right. We were going to put the patch on our website and direct people to it and then release it. Part of the --
Q. Would that be true if Apple objects?
A. Yes. Let me elaborate a little bit. Can I?
Q. Well, your counsel will have a chance --
A. Sure.
Q. -- to go into it more. But I wanted to find out if, in fact, you would have done it even if Apple had objected.
A. Yes, you often have to do this in order to be able to work in this area because everybody is going to object to having you release information, even if you do it responsibly.
Q. We'll get to that in a minute. But if you would go to Page 800-5. Now, Apple did not approve this publication, correct, that we're looking at, Exhibit 800?
A. They did not.
Q. And looking at Page 5 where it says "vulnerability

analysis"?

A. Right.

Q. That's describing the vulnerabilities that were found in the iPhone that enabled the hack, correct? If you go down at the bottom of that paragraph, it even identifies how you found the vulnerability, right?

A. Actually, what I was doing is identifying the software in which we found vulnerabilities, but it's not identifying the vulnerabilities.

Q. And it's saying, "The vulnerability we discovered and exploited was found in Mobile Safari using fuzzing," correct?

A. Correct.

Q. So you identified the method that was used?

A. Right.

Q. And what are attack scenarios underneath that?

A. Oh. So these are -- attack scenarios, basically when there's a vulnerability, you talk about when the attack happen. So in our example, we talked about since the iPhone has WiFi, which is wireless networking, if you were to go to a cyber cafe and get on that cyber cafe's WiFi network and the owner of that network was malicious, they could hack your iPhone. So we're describing different scenarios in which the attack is possible.

Q. And if you go to Page 800-6, there's a paragraph

entitled, "Black Box Exploitation."
A.   Right.
Q.   At the top it says: "Once a vulnerability has been identified, the next step is developing a functioning exploit." What is a "functioning exploit"?
A.   A functioning exploit is when the attack actually works. Let me differentiate between two phases. So the first phase of doing a security analysis like this is to find out where there's a vulnerability; for example, in this case, the RAM ghost effect, the index, all of these vulnerabilities that we talked about. You can stop there. That could be okay. Now, you know the thing is vulnerable, or you can take it all the way and actually build an attack that exploits that, and that would be the message that you guys saw.
Q.   And that was part of your security evaluation, as you called it, right?
A.   We did build the attack.
Q.   What is the paragraph entitled "black Box Shell Code Development" refer to?
A.   Well, when you attack a system, you have to write code that will run on that system. So after you discover a vulnerability and exploit it, the next step is to run the shell code. In the Haifa Report, the shell code was what piece that's in the I/O buffer that you have to jump into

later. In our case, we also wrote shell code to build an example exploit.

Q. If you could go to Exhibit 802, please. Is 802 a copy of the presentation Mr. Miller gave at the Black Hat conference on August 2nd, 2007?

A. Looking at the first few pages, I would say that's what it appears to be.

Q. And did Apple approve of this presentation?

A. No.

MR. STONE: I would move Exhibit 802 at this time, Your Honor.

THE COURT: Any objection?

MR. HAGAN: No objections, Your Honor.

THE COURT: Received.

(Exhibit No. 802 received in evidence.)

BY MR. STONE:

Q. Okay. If we could look at the first page, the talk was titled, "Hacking Leopard: Tools and techniques for attacking the newest Mac OS 10," right?

A. Right.

Q. And if I recall correctly, originally Mr. Miller was gonna do a talk on hacking Leopard, which was the new operating system, not the iPhone?

A. That's right.

Q. And you folks had obtained a prototype or beta copies of the Leopard software under a developer's agreement, correct?

A. Well, actually, that's what I thought in my deposition. And after you asked me this, I went back and talked to Dr. Miller. And he told me that while he did click through the agreement, he never downloaded it.

Q. Downloaded which agreement?

A. Leopard. He never downloaded the Leopard code.

Q. I'm sorry?

A. Dr. Miller told me that he never actually downloaded the Leopard code.

Q. Okay. But your testimony was that you had signed up under the developer's program so you get access to the code to hack it, right?

A. You asked me about that, and at the time I thought that he had done that, and I asked him, and he told me that he went to the site to download Leopard, clicked through the copyright agreement page, and then when he saw that he had to download it and burn it to a disc and he didn't have a disc drive that he could use to burn it; he didn't download it.

Q. But that was his intention, correct?

A. Yes.

Q. So he was going to pretend to be a developer so he could get the software so that he could hack it, unbeknownst

to Apple, right?
A. No, he was not going to pretend to be a developer. He was a developer, and he had a lot of reasons why he was interested in loading an early version of Leopard, including wanting to run it himself as his operating system.
Q. But his intent was to hack it and write about it?
A. When Charlie works at something, he's always very curious. He wanted to load it, and I'm sure he would have taken a whack at it because that's what he likes to do. But he didn't actually download it.
Q. Let's go to Page 8 of Exhibit 802.
This has two quotes from Apple marketing materials, correct?
A. Oh. My Page 8 is not that.
Q. Should say "Apple" at the top?
A. Oh, that's my Page 7.
Q. I don't know why that would be.
A. So there's a "7" here, but 802-8.
Q. The one at the bottom?
A. My bad. Um -- right.
Q. And was one of the reasons you hacked the iPhone to keep Apple honest in its representations in the marketplace?
A. Yes.
Q. Now, if you could go to Page -- I think it's 20 -- if you give me one second, I'll tell you -- Page 20.

Page 802-20, sir.
A. Right.
Q. Okay. It says, "They make exploitation fun." Who's the "they"?
A. I would imagine he was referring to Apple.
Q. And Mr. Miller is an employee of your company?
A. Yes, he is.
Q. And do you supervise his work?
A. I don't supervise him very closely, but he reports to me.
Q. You're the president of the company?
A. I am.
Q. And you have the ability to stop presentations like this if you want to stop them?
A. Yes, I do.
Q. Now, on the slide that says "They make exploitation fun," Mr. Miller revealed that Apple doesn't randomize anything. What he's talking about is they don't randomize certain portions of memory, correct?
A. That's correct.
Q. That makes it easy to hack the Apple product. Is that what he's saying?
A. Right. So basically one of the countermeasures that's commonly adopted for buffer overflow attacks is to take memory and randomize it, 'cause if you randomize the memory

and you're trying in our example to get to the index variable, you're not going to be able to go sequentially and get there. So this is a countermeasure in our cat-and-mouse game against hackers in security.
Q.   But Apple hadn't made that countermeasure at the time this was published, correct?
A.   What do you mean? In what?
Q.   In their products. Where he says Apple doesn't randomize, that's an indication they had not done it, correct?
A.   In some products. They did it in some.
Q.   Where Mr. Miller wrote, "The heap is executable," code can be executed in the heap, H-E-A-P, correct?
A.   The heap is in RAM.
Q.   And that's similar to the stack or RAM that we looked at for the ROM 3 card, correct?
A.   Right.
Q.   Now, if you could go to Page 23, please, of Exhibit 802 -- actually, I'm sorry -- if you go to Page 22, it will show the cover slide.
A.   Yes.
Q.   This is the page -- 802-22 is the iPhone details, correct?
A.   Right.
Q.   And then the rest of this talk gave the details of the

iPhone hack, right?
A. Right.
MR. STONE: If you go to the next slide, please.
BY MR. STONE:
Q. This slide is entitled, "How to find a Mac OS 10 0-day." Do you see that?
A. Yes.
Q. And a 0-day is a vulnerability that is not generally known, correct?
A. That's right.
Q. And a 0-day can also mean a vulnerability for which no patch has been issued, correct?
A. Right.
Q. And this slide shows one way to find a vulnerability in the Mac operating system, correct?
A. That's correct.
Q. So that Mr. Miller was saying, find some open source in the Mac product, read the change log for that software, find a good bug, and then, he says, "profit" exclamation point.
A. Right. This information here is not news to the Black Hat community. This is a standard way. This is kind of a set-up slide that he's setting up the rest of it.
And in a deposition I was asked about this word "profit." Did Dr. Miller want to profit from this? And afterwards I remembered that this is a reference to the

classic all-time security paper called "Smashing the Stack for Fun and Profit," which is the first paper to kind of show how a buffer overflow can work, and it's the one that we always teach from and cite. So he put the "profit" in this, speaking to an audience of people who have read this paper, to try to get a laugh. He wasn't actually looking to profit.

Q. Didn't you say at your deposition that you wished Mr. Miller had not put that in the presentation?

A. That's right. Like I said, after my deposition I remembered that that referred to that paper.

Q. But at your deposition you didn't recall that from the time that Mr. Miller gave his report, right?

A. That's right.

Q. And that was in August 2007?

A. The deposition was in October, I think.

Q. No -- and the presentation was in August.

A. Right.

Q. So a few months apart?

A. Right.

Q. Now, you can't guarantee that no malicious hacker would follow these steps to exploit a vulnerability in the Leopard or Mac's OS X operating system, correct?

A. I can pretty much guarantee that they would. This is the standard way that people do it.

Q. Let's go to Page 802-25, please.
And again, this is a presentation that Apple did not want you to give, right?
A. I don't know that. First, I didn't give it, but you're referring to ISE -- I'm assuming that he's referring by "you" to plural, my company, as opposed to me. There were people from Apple in the audience, and Dr. Miller told me that after his presentation they came up and had a friendly conversation, so I don't know that Apple necessarily was upset. Their iPhone was more secure when we were done.
Q. Let's talk about that. Look at Page 802-25. This is entitled "The Vulnerability."
A. Right.
Q. And this is the process where Mr. Miller explains how one could use the heap to overflow and write an exploit to hack the iPhone, right?
A. Right.
Q. And he gives a number of total bytes that can be overflowed, correct?
A. That's correct.
Q. And he discloses that he found it the old-fashioned way, fuzzing, correct?
A. That's right.
Q. If you go to the next page, 802-26.
Do you know what Mr. Miller meant by his title at the

top? "Another change log entry, another Safari 0-day"?
A. Let's see. Change logs are things in software that will show you when things change. It's what it sounds like. It's a log of a change. And they're often used by hackers to figure out what the bug was that caused that change. If you get your access to the change logs for a product, you can then pretty easily figure out how to attack previous versions of the product. And then people who haven't upgraded to the latest release are going to be vulnerable to that.
Q. Where it says "Another heap overflow" and it has some script language, that's Mr. Miller of your company giving an example of an unchecked input that overflows memory in the iPhone, correct?
A. Right.
Q. Go to the next page if you will, please, Page 27 of Exhibit 802.
A. Okay.
Q. This is entitled, "Black Box Exploitation of the iPhone." What is a black box exploitation?
A. So the point of this slide is that you -- when you go to attack something -- sometimes Mr. Mordinson, for example, had access to the ROM code. So we call that a white box evaluation because you get all of the code in the system. So it's a lot easier because when you have the code, you can

figure out where all the bytes go, and you can perform your attack.

We only had an Apple iPhone. We had to treat it as a black box where we couldn't see inside it. We didn't know what it was doing, so that was a high-level description of the kind of attack that this was.

Q. And then you gave the details of the exact process, correct?

A. I'd have to look at the slides to see if he gave all the details.

MR. STONE: Let's look at Page 802-30, please.

(Document displayed.)

BY MR. STONE:

Q. This is entitled, "A good" -- and "good" is in quotes -- "crash." Do you see that?

A. Yes.

Q. What Mr. Miller meant by that is by causing the iPhone to crash, you can obtain data that can be used to hack it, correct?

A. That's right.

Q. And --

A. Some crashes give you no information. And what he was saying here is that this particular crash was useful to him.

Q. In this publication that Mr. Miller made, he described how to cause the phone to crash, correct?

A. That's right.
Q. And what do you understand Mr. Miller to have meant at the bottom where he said "old school heap overflow"?
A. What he was kind of -- this was very much a tongue-in-cheek presentation where he was preaching to the choir at Black Hat, and he was basically saying that nothing is fancy here. This isn't any new technique. We're using heap overflow.
Q. And did he have Apple's permission to disclose that vulnerability prior to this?
A. At this point, Apple had already acknowledged us and issued a patch.
Q. But they had not approved him disclosing these details; isn't that true, sir?
A. That's right.
MR. STONE: Now, go to Page 802-31, please.
(Document displayed.)
BY MR. STONE:
Q. This is entitled, "Controlling the inputs," right?
A. That's right.
Q. And this gives some pretty specific details that would give one the ability to write four bytes anywhere in the memory of the iPhone, correct?
A. Right.
Q. And then if you go to the slide at Page 33 entitled,

"Getting PC"?
A. Okay.
Q. What does that mean, "Getting PC"?
A. PC is the program counter, and that shows you where you are in the program.
Q. Is that similar to an index variable?
A. No.
Q. Is it similar to a stack pointer?
A. Yes.
Q. It says, "We chose to overwrite a saved return address on the stack." What does that mean?
A. Okay. So I guess I'm not gonna get away with not telling you what a stack is. Sorry.
When a program is in memory in a computer -- or in a device for that matter -- the program is composed of subroutines and functions. These are -- think of them as modular pieces of code that do one thing.
So, for example, let's say that you're gonna write a program to calculate, you know, the trajectory of an airplane, and one of the things that you need to be able to do is compute a factorial. That's a function that you need. So instead of embedding all the code for the factorial, every place that you need the factorial function you're just gonna write a factorial function, and you're gonna call it whenever you need it.

Now, the way the computer works is, when you call a function when you're running, say, okay, it's time -- I need a factorial, call factorial, it takes what's called a stack frame. It takes a bunch of memory and allocates it on the stack. And the stack is just a place in memory where you're going to put some things every time you call a function.

And a computer is a very simple device. It just does what it's told.

And after you call the factorial, now you want to go back to what you were doing before you called the factorial. How are you going to know where to go in the program to get back to where you were before you called the factorial? One of the things you put in the stack frame in memory is a return address. That's the address in memory right after where you previously were before you called the function.

And so factorial runs, gets the result, and then looks in the return address to see now where do I need to go back to.

Now, a common technique when you're hacking is to say, "Let's write a function." It will get put on the stack, and if one of the variables or buffers in that function -- say that factorial has X and Y, right? If X doesn't have bounce checking on it, you can write a really long value into it -- just like I did earlier -- and start overriding frame pointers on the stack. And you can change the return address back to your shell code instead of going back to the program and doing legitimate things.

Q. So was Mr. Miller acting maliciously in using this shell code and a return address on the stack?

A. No.

Q. Now, when you overwrite a safe return address, that's not the normal function of the program, is it?

A. That's a hack. That's what hackers do.

Q. And this slide goes on to reveal that it was found by fuzzing from SP. Does that mean "stack pointer"?

A. Yes.

Q. What does that mean where it says "setting the other register to a stack value"? That is where you want it to return to your shell code?

A. I'm not sure what he was doing with the other stack value.

Q. And those are instructions below that provide detail on how to do that on the Apple iPhone?

A. I think that this is data. It's a dump of the crash log.

Q. Which would help you in hacking the iPhone?

A. Right.

Q. And then executing on the stack down below. Do you see that?

A. Right.
Q. That's disclosing that code is executable, the stack, which is similar to the RAM in the ROM 3 card, right?
A. Right.
MR. STONE: Now, go to Page 34, please.
(Document displayed.)
BY MR. STONE:
Q. By the way, did the Nipper attack also overwrite part of the stack?
A. Yes, it did.
Q. And it had a shell code that had a return address on the stack?
A. No, it didn't use a return address in the same sense that most programs do. But it had an exception handling address which pointed back to the middle of the stack.
Q. Now, go to Page 34 where it says, "Search for your shell code." What is shell code? Is that the malicious code?
A. Shell code is the code written by the attacker to do whatever it is that the attacker wants to do.
Q. And did the Nipper attack have shell code in it?
A. Yes. The Haifa and the Nipper -- the Haifa put the shell code in the I/O buffer, and Nipper put the shell code on the stack.
MR. STONE: Now go to Page 38 of

Exhibit 802, please.

(Document displayed.)

BY MR. STONE:

Q. Is this -- this is entitled, "Make it happen," right?

A. Right.

Q. Is this the actual exploit code?

A. I believe this is part of it.

Q. And so your company disclosed specific strings of code that could be used to hack the iPhone in its presentation at the Black Hat conference, right?

A. No. They presented code that could be used to hack the previous version of the iPhone; but by the time this was presented, the iPhone had been patched. This would not have worked anymore.

Q. You've testified that you cannot guarantee that no malicious hacker could use the methods and steps in this presentation to create some new hack of iPhone, right?

A. Right. They'd have to make their own attack.

Q. Well, you can't guarantee that they wouldn't use these methods and steps to do that. Isn't that what you testified to, sir?

A. They wouldn't use these exact steps, but they might increase their knowledge through looking at this and be able to develop hacks.

Q. And your company was the first to hack the iPhone and

publish the results, right?
A. Again, there were people who knew how to unlock it but not how to run shell code on it.
Q. So is it your testimony that your company would not be responsible for every hack of the iPhone that comes after your publication of the details of the hack?
A. That's right.
Q. And it's your testimony you're not responsible for every hack of the iPhone after this presentation because whether you had done it or not, it all would have happened, correct?
A. I'm certain that this would have happened.
Q. And that's because the iPhone had vulnerability in it, right?
A. That's correct.
Q. Now, it's your testimony that the details in this presentation would not allow somebody to create a hack of the iPhone once it was patched, correct?
A. That's right.
Q. So it's your testimony that once Apple patched the iPhone, the details could not be used for any hacks in the future?
A. Yeah.
Q. Now, if for any reason Apple chose not to issue a patch, do you believe your company would be responsible for

all future hacks of the iPhone?

A. No. We were gonna issue the patch, but I think it would have been very irresponsible of them not to issue a patch for this.

Q. How was Apple able to issue a patch for the iPhone?

A. They have a mechanism: When you synchronize your iPhone with your computer, it will download any patches that have occurred to the phone.

Q. You're aware that patch codes for the EchoStar access cards can be delivered through the satellite signal, correct?

A. I am aware of that.

Q. And you agree that the attack that was posted in the Nipper posting in December of 2000 would not be possible unless the buffer in the ROM 3 card could be overflowed in the first instance, right?

A. If the buffer could not have been overflown, then that particular attack would not have worked.

THE COURT: Tell us when is a good time for a break. Now?

MR. STONE: Yes.

THE COURT: You're admonished not to discuss this matter among yourselves nor to form or express any opinion.

Take a recess. We'll come and get you in about 20 minutes.

(Recess held at 3:00 o'clock p.m.)

(Proceedings resumed at 3:18 p.m.)

(In the presence of the jury.)

THE COURT: We're back in session. All parties are present. The jury is present.

This is still cross-examination by Mr. Stone on behalf of NDS.

MR. STONE: Thank you, Your Honor.

CROSS-EXAMINATION (Resumed)

BY MR. STONE:

Q. So, Dr. Rubin, backing up for a second, it's clear that the Nipper posting in December of 2000 would not be possible unless the buffer could be overflown in the first instance, right?

A. That's right. The Nipper posting required the ability to overflow the buffer.

Q. And the black box pirate device that EchoStar and NagraStar acquired also used a buffer overflow attack very similar to the Nipper attack, right?

A. That's right.

Q. And the black box device would not work unless the buffer could be overflowed in the first instance, as well, correct?

A. Is that what you just asked me?

Q. I'm talking now about the black box.

A. Right.
Q. The black box device would only work if the buffer could be overflown in the first instance?
A. That's right.
Q. And you testified that buffer overflows are very easy to fix and easy to prevent, correct?
A. In general they are, yes.
Q. And you testified in this case that it would take two assembly language programming statements to protect one communications buffer against an overflow, correct?
A. That's right.
Q. And the ROM 3 card had one communications buffer, correct?
A. Correct.
Q. And you were asked at your deposition, don't you recall, that if you were asked to write code to prevent buffer overflow for the ROM 3 card, how would you write that code? Do you recall that?
A. I remember that, yes.
Q. And didn't you say you would write code so that when the buffer was being read in, you would simply count the bytes, and when it got to the length of the buffer, it would stop reading?
A. Right. That's what I would do.
Q. And it was your testimony that that code that you just

described would effectively prevent anyone from using a buffer overflow attack; is that correct?
A. From using that particular buffer overflow attack, yes.
Q. Well, from overflowing that buffer that's protected by that code is what you testified to.
A. Right.
Q. So there could be no attack on that buffer if it had those two lines of code that counted bytes that you just described, correct?
A. That's right.
Q. And you testified, didn't you, that you have no reason to disagree with Mr. Jones' opinion that the patch code used on the ROM 3 card effectively completely precluded the buffer overflow vulnerability from being used?
A. That's right.
Q. You also testified you had no reason to dispute Mr. Jones' testimony that the patch code for the ROM 3 card actually checked twice for buffer overflow and precluded it?
A. That's right. Although after listening to Christophe Nicolas today, I think it's much more complicated than that.
Q. Well, you had plenty of time to study this issue from when you were retained in July of 2007 and March 26, 2008, when you saw the source code, didn't you?
A. Right. Except that --

Q. Didn't you?
A. Well, that wasn't what I was tasked to do.
Q. Nobody asked you to look at that; is that right?
A. No.
Q. And when you came out and looked at the source code, did you even go and look at the patch code?
A. No, I did not.
Q. Did somebody tell you not to?
A. No.
Q. Now, you've testified that the Nipper posting would only be possible if the buffer could be overflown in the first instance, right?
A. Right.
Q. So if the ROM 3 cards had been electronically patched before that posting, then the Nipper posting could not have been used to dump the contents of the ROM 3 card, correct?
A. It depends what the electronic patch did.
Q. If it prevented buffer overflow?
A. Then the cards that received that patch would not be vulnerable to that attack.
Q. And that would be true also of the black box attack as well, correct?
A. What would be true -- are you asking me if the black box would not be able to hack cards that had that patch on it already?

Q. Yes, sir.
A. That's correct.
Q. Is it -- if your company had been advising Kudelski, you would have recommended immediate development of a patch upon discovery of the buffer overflow vulnerability; isn't that correct?
A. That's correct.
Q. Now, you saw evidence that Kudelski and Nagra were aware of the buffer overflow vulnerability prior to October 2000, correct?
A. Not the vulnerability. I saw they were -- I saw evidence that they knew they were not checking the buffer bounds, but not that they considered that it was a vulnerability.
Q. Well, the black box pirate device -- you used a buffer overflow vulnerability, didn't you?
A. Right.
Q. And Kudelski had that no later than October 2000, right?
A. Right. That's what I heard.
Q. You don't know of any reason, any technological reason, why a patch could not have been issued to the ROM 3 cards at that time, correct?
A. That's not correct.
Q. Well, let me -- I'll show you your deposition.

A. Let me say this was based on yesterday's testimony where I heard information I was not previously familiar with.
Q. Well, as of March 27th, 2008, your sworn testimony was you weren't aware of any technological reason that a patch could not have been issued at the time the back box was analyzed, correct?
A. I was not aware of it then. And I am now.
Q. Well, had you bothered to talk to anyone at the client prior to testifying twice in this case under oath?
A. No, I had all the information that I needed to form my opinions in this case about one piece of code deriving from the other, and that is outside of that scope.
Q. When was the first time you ever spoke to anyone at the client?
A. I think it was the Saturday before -- no -- the Monday before the trial started.
Q. Did anyone tell you that your testimony was incorrect and that you should have changed it?
A. No. I realized when I was listening yesterday that I had not understood everything about the constraints that the site designers were operating under.
Q. It is your testimony yesterday that you did not see any evidence from your review of the evidence explaining why Nagra or Kudelski did not issue a patch after the black box

was obtained and before the December 2000 posting, correct?
A. Right. I didn't understand at that point the resource constraints that they were operating under. And so I, you know, without that information that I now have, I was -- that was my opinion then.
Q. Didn't you say that your company, had it been advising Kudelski, would have recommended fixing the buffer overflow vulnerability before the December posting and after the acquisition of the black box?
A. But the December posting and after -- the black box was acquired before the posting, right?
Q. Right?
A. I don't know what you mean before the December posting and after the back box acquisition.
Q. You had advised Kudelski to fix buffer overflow vulnerability as soon as they acquired the black box, correct?
A. I would have advised that, but they would have explained to me why that's a lot harder than I realize, and I would have understood that explanation.
Q. Now, the resource constraints you testified to applied only to the ROM 2 card, correct?
A. No.
Q. Was there room in the EEPROM for patch code to prevent a buffer overflow on the ROM 3 card?

A. I don't know.
Q. Now, a patch was issued in this case after the December 2000, posting?
A. Right.
Q. And you're saying that could not have been in October 2000.
A. I haven't said, but I'm saying.
Q. My question is, is it your testimony that patch could not have been issued in October of 2000?
A. No, I didn't say that.
Q. Well, isn't it true that if a patch had been issued that prevented buffer overflowing before the Internet postings, the memory aliasing feature in the ROM 3 card could not have been exploited?
A. Not with that attack. I don't believe that it's true that it could never have been exploited.
Q. How else do you exploit memory aliasing unless you overflow the buffer?
A. You -- the memory aliasing would happen anytime an address was accessed that was above the legal terminating address for the memory. And that might have happened any number of ways. This particular attack required overflowing the buffer first, and so that attack wouldn't have been possible.
Q. Now, isn't it true that if the patch to prevent buffer

overflow had been deployed prior to the postings, the ROM 3 card would have been a more secure and better product?
A. Yes.
Q. And wasn't that the reason your company hacked the iPhone, to make it a more secure and better product, because it forced Apple to patch the buffer overflow vulnerability?
A. It's one of the reasons. We already discussed the publicity and, you know, educating security people, but that was one of the reasons.
Q. Isn't it your claim that your company hacks products and finds vulnerabilities to serve the greater goal of having vendors fix the vulnerabilities as soon as the vulnerabilities are disclosed to them?
A. I would say that that's too strong a statement. I think it's -- we do it so that they can fix the vulnerabilities as soon as they can.
MR. STONE: Your Honor, I would like to play from Mr. Rubin's deposition at Page 172, Lines 12 through 18.
THE COURT: Beginning question, "Well"?
MR. STONE: Yes, sir, at Line 12 to 18.
THE COURT: That's denied, Counsel.
MR. STONE: Thank you, Your Honor.
BY MR. STONE:
Q. Mr. Rubin, did you testify that when you hacked the iPhone, it was to serve the greater goal of having vendors

fix the vulnerabilities when the vulnerabilities -- didn't you testify that it was for the reasons of greater disclosure and fixing of vulnerabilities that the xbr21 disclosure might have led to a better product?

A.   Can you say that again, please.

Q.   Sure. Didn't you testify that having vendors fix vulnerabilities as soon as they are disclosed was a reason why the xbr21 disclosure might have led to a better product?

A.   I'm assuming I said that in my deposition.

Q.   Do you believe Nagra Kudelski as the vendor of the ROM 3 card had a duty and responsibility to patch the buffer overflow vulnerability as soon as they became aware it was being used for pirate attacks?

A.   I think they should have done it as soon as they could. And there are some constraints on the card that would make that really difficult to do very, very quickly.

Q.   Do you know when they first began work on designing and developing the patch?

A.   I believe that Mr. Nicolas said that they did that after they analyzed the black box and met with ST Thomson and figured out what the problem was.

Q.   Now, earlier you testified that the Headend Report, or the Haifa Report, as you call it, and the Nipper posting have the same DNA?

A.   That's right.

Q. But you previously testified that the comparison you made showed that the structure of the attack was similar, correct?
A. The structure, yes.
Q. That's what you testified to earlier today, correct?
A. Right.
Q. Now, you and I have the same physical structure, don't we?
A. Yes.
Q. I have two arms; you have two arms?
A. Yes.
Q. I have two legs --
THE COURT: Counsel, counsel...
BY MR. STONE:
Q. You and I don't share the same DNA, correct?
A. That's right.
Q. And did you compare the DNA between the black box and the Nipper posting?
A. I did.
Q. And was it your opinion that the Nipper and the black box shared the same DNA?
A. Yes.
Q. And if I could show you your report, Exhibit 799, at Page 32 of 38. It's -- 799-32 is the bottom number?
A. I'm there.

Q. Okay. Now, in your report, you agreed with Mr. Jones that as between the Nipper posts and the Headend Report, the programming methodologies and styles are different, correct?
A. That's right.
Q. The programs are different lengths, correct?
A. That's right.
Q. The hexadecimal byte sequences are different, correct?
A. That's right. They would have to be if they were different programs.
Q. Different addressing methodologies are used, correct?
A. Where does it say that?
Q. Right after the byte sequences are different.
A. How far down the page?
Q. About the fourth line.
A. Right.
Q. And then you agreed that many other specific aspects of the two programs are different, correct?
A. That's right.
Q. Now, when you say "the programming methodologies are different," what is a programming methodology?
A. So in this particular example, the Haifa Report, you know, there are a lot of ways -- a methodology is just how somebody goes about writing code. In this case, one example I could use would be the using -- developing your own code for certain functions that built-in functions exist for and

then otherwise using a built-in function.
Q. You also said the programming styles were different, correct?
A. That's right.
Q. And in what way were the programming styles different?
A. It's kind of a hard thing to describe somebody's programming style. It's like asking someone what their writing style is like and how it's different. But I will say that the Nipper code is much more compact, more cleverly crafted.
Q. And then you also agreed, right, hexadecimal byte sequences are different?
A. Right.
Q. What did you mean by that?
A. That it's not a carbon copy of the other one.
Q. And did you ever measure the degree to which there was a difference in the hexadecimal byte sequences?
A. No, I did not. I don't see that that would be relevant at all.
Q. You said that different addressing methodologies are used?
A. Right.
Q. And is it true that Mr. Mordinson wrote his own, while the Nipper posting uses a library call function?
A. Right. That's what I was calling a built-in function.

Q. You said many other aspects of the two programs were different, correct?
A. That's right.
Q. Would that include the Headend Report as the shell code in the communications buffer while Nipper places it in the stack?
A. Exactly.
Q. And can you place the same number of bytes in the communications buffer as you can in the stack in the ROM 3 card?
A. No.
Q. What is the difference?
A. So in the communications buffer, you can only put a hundred bytes. And I don't recall the size of the stack. But it's -- I remember it's not a hundred.
Q. Now, the Nipper posting uses a routine built into the card to transmit data out of the card, correct?
A. That's right.
Q. And Mordinson wrote his own program, correct?
A. That's right.
Q. And the Nipper posting terminates by jumping into the main processing loop, correct?
A. That's right.
Q. And how does Mordinson's code end?
A. It goes into an infinite loop.

Q. And didn't you testify that the Headend Report would not have been sufficient to create the xbr21 posting?
A. There were -- what I said was that there were other aspects about Nipper that didn't seem to be known or weren't utilized by Mordinson in his code.
Q. Didn't you testify that the Headend Report would not have been sufficient to create the xbr21 posting?
A. I don't think I said that.
MR. STONE: Your Honor, I would like to read Page 21, Lines 7 through 12.
THE COURT: Overruled. You may not.
BY MR. STONE:
Q. Is it correct, Dr. Rubin, that there was programming code in the xbr21 post that was not in the Headend Report that NDS did not know was necessary for the attack to be functional?
A. I'm not sure that NDS didn't know it. I don't know that they put everything that they knew into their report, but there was code in the Nipper posting that was not in the Haifa -- in the Headend Report.
Q. Well, if somebody had used the code in the Headend Report at the end of the program, it would simply hang, correct?
A. That's right.
MR. STONE: And now if we could look at

Exhibit 511A, the Nipper posting?
This is in evidence.
(Document displayed.)
BY MR. STONE:
Q. Now, if you go to the second page of 511A, sir, in the third line from the end of the code, there's --
A. Are you counting the incorrect checksum as a line?
Q. Yes.
A. So the first complete line from the bottom?
Q. Where it says "Ox73, Ox81."
A. Right.
Q. Is it -- that's the so-called "73, 81 jump" identified by Mr. Jones?
A. That's correct.
Q. And that's an address in memory that you need to jump to, to continue to have the program function, correct?
A. That's right.
Q. And as part of that jump, the xbr21 code also passes parameters, correct?
A. Right.
Q. And could you agree that the 73, 81 jump is nowhere contained in the Headend Report?
A. That's correct.
Q. And you saw no evidence that anyone at NDS had figured out the 73, 81 jump that is contained in the posting?

A. I saw no evidence of that.
Q. And the 73, 81 jump is nowhere contained in the Headend Report, correct?
A. Yeah. I thought that's what you just asked.
Q. And so NDS' Haifa lab had the ROM code, but the Haifa lab never derived this 73, 81 jump for the passing of the parameters that are in the xbr21 posting?
A. I don't know. They didn't put it in the report. That doesn't mean they didn't derive it.
Q. Did you see any evidence in any of the documents or testimony that they had derived that?
A. No.
Q. And it was your hypothesis that xbr21 was able to derive the 73, 81 jump and the passing of parameters by having access to a dump of the ROM code, correct?
A. That would be one way to get it.
Q. And it was your testimony that it would have been difficult for somebody to figure out the parameters of the 73, 81 jump, correct?
A. Yes.
Q. And I believe you testified you saw no evidence that NDS figured out that jump, and you don't know how xbr21 was able to derive that jump as part of the posting on the Internet, correct?
A. That's right.

Q. So if I understand it correctly, in addition to these differences in the code that we have talked about, the xbr21 posting had code that NDS did not know about, you saw no evidence that they had figured out, and was necessary for a functional buffer overflow attack, correct?
A. No. The 73, 81 is not necessary for a functional buffer overflow attack. If the card goes into an infinite loop at the end, as it does in the Haifa Report, you still dump the EEPROM and have the ability to hack the card. This is simply a way that's a little cleaner, which is why it's my opinion that this is an improvement over the Haifa after more information was learned.
Q. Information that you don't know how xbr21 derived, correct?
A. No. Access to the ROM code would have been one way to do that.
Q. Now, let's take up the similarities that you talked about.
First, the buffer overflow. That's a vulnerability that's in every ROM 3 card, correct?
A. That's right.
Q. And that would be every unpatched.
A. Yes, sir.
Q. And that's the weakest link in the card's security.
A. I don't know about that. I think taken together, all

of the factors are a weak link.
Q. And the only way you can exploit those other factors is to have buffer overflow in the first instance, correct?
A. Every single one of them is necessary for the attack to work, so I don't see that one of them is more important than the others.
Q. Can you exploit the memory alias, then, if you cannot overflow the buffer?
A. No.
Q. And the buffer overflow vulnerability can be determined by sending invalid inputs through the process of fuzzing, correct?
A. Right.
Q. And buffer overflow vulnerabilities are well-known and common, and it's one of the first things an attacker would think of, correct?
A. That's right.
Q. Anyone who had access to the ROM and disassembled it would see that Nagra had deliberately failed to check for overflow in the communications buffer in the ROM 3 card, correct?
A. I disagree with that.
Q. Let's talk about memory aliasing. That's a property of the particular ST microchip that was used in the ROM 3 card, correct?

A. That's right.
Q. And did that exist in the ROM 2 card as well?
A. I don't know.
Q. How do you perform a buffer overflow attack on the ROM 3 card without using memory aliasing?
A. You just send it more than a hundred bytes.
Q. And didn't you testify that you did not know whether there was publicly available documents for the family of chips using the EchoStar access cards that references memory aliasing?
A. Yes, I don't know what all the documentation is for it.
Q. And you're aware that Mr. Jones looked at the data sheet for this family of chips, correct?
A. That's right. I looked at it as well after my first deposition.
Q. And you didn't dispute Mr. Jones' opinion that the data sheet indicated there was memory aliasing?
A. I didn't see memory aliasing in the data sheet.
Q. You didn't dispute Mr. Jones' opinion that it discloses memory aliasing, did you?
A. No. Before I looked at the time data sheet, you had asked me if I had any reason to dispute Mr. Jones' statement that there was aliasing. And at that point I didn't have any reason to dispute it, but I do now.
Q. So if I understand correctly, are you saying that if

your company used fuzzing to overflow the ROM 3 buffer, the bytes would not memory alias?
A. Well -- what?
Q. If your company were to use fuzzing to overflow the buffer --
A. Right.
Q. -- would those bytes memory alias?
A. Sure.
Q. Why would that be?
A. Because that's what the card does when you send it too many inputs.
Q. And that's because that is a feature of the card itself?
A. It's a property of the card.
Q. Property of the card?
A. Yes.
Q. So anyone overflowing the buffer would end up utilizing memory aliasing, correct?
A. That's right. I don't know about utilizing it, but they would cause it.
Q. And they would become aware of it, correct?
A. No.
Q. Well, didn't you testify that you can test very simply for memory aliasing?
A. If you're looking for it. You wouldn't probably

discover it on your own if you were just fuzzing it.
Q. Well, what would the test be to determine memory aliasing that you said would be easy to devise?
A. This test assumes that you know about memory aliasing and that that's what you're looking to verify whether or not it exists. So the test is that you would send it increasing-sized messages and you would have some way of determining that you were overriding memory in the lower regions of the card. They're many ways that you can do that, but you're not likely to discover memory aliasing that way unless you know that you're looking for it.
Q. Well, if you gave it escalating inputs and you saw whether you got an error message or if you started to see the same values again, that would tell you whether memory aliasing was occurring, wouldn't it?
A. Again, if you started out with the problem of asking yourself, "Does this card do memory aliasing," then, yes, that would confirm or not for you that that happened. But just because you saw that wouldn't necessarily lead you to the conclusion that there was memory aliasing.
Q. One of the other elements you identified was the incorrect checksum creating an exception. Do you recall that?
A. That's right.
Q. And at your deposition you testified you could not

identify any other way to create an exception in the ROM 3 card, correct?
A. That's right.
Q. And didn't you testify also there was only one way you could think of to get shell code to a place where it could execute on the ROM 3 card?
A. Yeah. But I also since changed that opinion with new information about the card.
Q. After March 27th, 2008?
A. Right.
Q. Now, the use of the index variable -- can you envision a buffer overflow attack that does not use or make use of the index variable?
A. Yes.
Q. And what would that example be?
A. Any buffer overflow attack that isn't crafted the way these three that we've looked at are crafted, you would overflow the buffer but not put in so many bytes that you change the index variable. So say put 115 bytes in and you've overflown the buffer, but you haven't touched the index variable.
Q. And would that give you the proper return address for the shell code?
A. No. You were asking me of an attack that didn't change the address, but you didn't say that I was trying to do what

this attack was trying to do.
Q. Now, when hacking the iPhone, Mr. Miller of your company took advantage of certain vulnerabilities in the iPhone, right?
A. That's right.
Q. One of those is the fact that the iPhone uses a heap or stack that allows the execution of code, correct?
A. That's right.
Q. Another was the fact that the iPhone was subject to a buffer overflow, correct?
A. Correct.
Q. He also used shell code in conjunction with the buffer overflow to get the iPhone to do what he wanted, correct?
A. Right. And the things that you're describing right now are basically the -- the basic components of any buffer overflow attack commercially in the field. Microsoft pretty much once a week releases patches to their operating systems and applications that are designed to address exactly these kind of issues. So these are very common things. They're not specific to the iPhone.
Q. Mr. Miller also found out that there was no randomization of memory in the iPhone, correct?
A. That's correct.
Q. And then he used filler bytes to overflow the stack in the iPhone, correct?

A. That's right.
Q. Did I miss anything that he utilized in the attack?
A. Let me think. I would guess you did, but I don't -- I'm not familiar with the attack at the level to say what they are.
Q. So if a subsequent attack on the iPhone exploited those same vulnerabilities, would you conclude that Mr. Miller must have been the source of that attack?
A. I don't think that would give me enough information to come to that conclusion. I would look at their attack and I would look at his attack the way I looked at Haifa and Nipper and saw the same DNA and same properties, and these were all things not known. Whereas, all the things in the iPhone were pretty much known, then I would conclude that.
Q. And Mr. Miller was giving a presentation because all of that was already known?
A. The particular things that you described, that the stack was executable, that there was a buffer overflow vulnerability -- take that one back -- I'm not sure if it was known, but anyone that would have fuzzed on the iPhone would have known that there was a buffer overflow vulnerability. And the other things you mentioned were known.
Q. The four items or key components you identified earlier -- the buffer overflow, the memory aliasing, use of

the index variable, and the exception from an incorrect checksum -- you testified all could be deduced by an appropriately skilled individual who had access to the ROM code, correct?
A. What I said was that an individual at least as skilled as David Mordinson, spending six months and under the guidance of someone like Zvi Shkedy who had broken up the ROM, with the ability to ask that person questions could, in fact, given the ROM code, deduce those vulnerabilities. In fact, he did it.
Q. And you also testified that there were people out there equally capable as Mr. Mordinson?
A. That's right.
Q. Now, you disagreed with Mr. Jones' opinion that the buffer overflow vulnerability was either intentional or due to incompetence, correct?
A. Before I had looked at the code, I disagreed that that buffer wasn't checked intentionally. And now that I've looked at the code, I believe that they knew that they weren't checking the bounds and that they had good reason not to check the bounds.
Q. And didn't you testify that whether there is memory aliasing or not, one should always check for buffer overflow?
A. Yes, that's correct.

Q. And in reviewing the code, did you find that there were two other buffers in the program for the ROM 3 code?
A. I did.
Q. Were those checked to make sure they did not overflow?
A. Yes.
Q. Could those two buffers be used for a malicious pirate attack?
A. I don't believe so.
Q. The only buffer that can be used for a pirate attack is the communications buffer, correct?
A. That's right.
Q. And at the time you testified originally, you had not seen the source code?
A. That's right.
Q. And your testimony originally was that it would be unbelievable that somebody would deliberately not check for overflow, correct?
A. That's right.
Q. And did anyone at the client tell you before you gave that opinion that it was not true that they had deliberately failed to check?
A. As I mentioned earlier, I had not talked to anyone at the client.
Q. Did you disclose your opinions to the attorneys for the client before you testified?

A. No. That was a bit of an unusual question, so I think the attorneys became aware of it in my deposition when you asked it.
Q. And without having seen the code, your original testimony was that from looking at all the evidence, the conclusion you drew was they simply were not thinking about buffer overflow attacks, correct?
A. Right. Before looking at the code, I was under the impression that someone that wouldn't check the bounds on an I/O buffer was just not thinking about it.
Q. And Mr. Jones, before seeing the code, had concluded from the design decisions he had seen that it was more likely than not that it was a deliberate decision, correct?
A. He did. I totally disagree with that, but that's what he said.
Q. And it turned out he was correct, at least with respect to it being a deliberate decision, correct?
A. That's right.
Q. Now, did you do anything to examine, inspect, analyze, or verify the reason given in the code for the deliberate failure to check for overflow?
A. Yes, I did.
Q. And didn't you testify that if we assume that the same author wrote the code that explicitly checked with the two buffers that cannot be used for piracy and deliberately

chose not to check for the one buffer that could be used for piracy, your explanation would be that the person maybe wrote one section of the code in a good mood and one section of the code in a bad mood, correct?

A. No. I did say that, but not to that. You asked me how could it be that the comments -- I said that the comments in one place were written in a particular style, and in another place they were written in another style, and that was one of my reasons for speculating that perhaps they were written by different people. And you asked me if they weren't written by different people, if they were written by the same person, how would I explain that, and I said that maybe they were in a good mood when they wrote one part of it and a bad mood when they wrote the other part.

Q. And the bad mood would be not checking the buffer part of it?

A. I said that the comments in one section were kind of dull, very much like a computer programmer who's just not, you know, very much creative or something; and the other place he was saying "hanky-panky" and things like that. And I said, "I think it's a different person."

And he asked me, "Well, what if it's" -- you know, "What if it's the same person?"

And I said, "Well, they were in a different mood."

That's what the conversation was.

Q. The hanky-panky comments in the patch code?
A. Right.
Q. That's not the code wherein there is a deliberate failure to check for the communications buffer, correct?
A. Right. No, you're right.
Q. And you didn't look at the patch code at the time you gave your deposition, right?
A. Right. I was drawing that statement from the testimony yesterday where you heard "hanky-panky," and I was using that as an example.
Q. I was focusing on your deposition testimony after you had looked at the source code where you determined the author failed to check for buffer overflow in the communications buffer.
A. Fair enough. But my point is still that the commenting was written in a different voice in the two sections.
Q. Which was the bad mood? The failure to check?
A. I don't remember.
Q. Now, one of the explanations given for the failure to check the communications buffer or overflow was the claim that there was no memory after the buffer, correct?
A. That's one of them, yes.
Q. And if it turned out that the author of the code actually knew about memory aliasing, that false explanation would be indicative of an insider attack, correct?

A. No, I think the insider attack theory is fatally flawed, and I've heard -- I've heard that theory before from Mr. Jones. But I have an explanation of why that can't be possible or at least is incredibly unlikely.
Q. Did you ever speak to the author of the code, Mr. Osen?
A. No, I did not.
Q. Do you know whether Mr. Osen was aware of memory aliasing when he wrote the code?
A. I believe he was not aware of it.
Q. Do you know for a fact that --
A. I do not.
Q. Going back to your report, Exhibit 799 --
A. Yes.
Q. -- and focusing you, please, on Page 36 of 37. And I believe it has 799-36 at the bottom.
A. Right.
Q. And that was signed by you on August 29th, 2007, correct?
A. That's right.
Q. And you wrote, "This report is based on the evidence provided to me in this case," correct?
A. Yes.
Q. And that evidence was provided to you by counsel, I take it?
A. That's right.

Q. And you reviewed those materials and tried to be as accurate and thorough as you could be in preparing your report, right?
A. Of course.
Q. And you stand by everything that's in your report, correct?
A. We discovered in my deposition, and you were nice enough to point out, a couple of minor mistakes.
Q. Those were the diagrams that were wrong?
A. Right. One of the lines in the Haifa report I had drawn in the wrong place. And then the other was not really a mistake, but you pointed out one of my conclusions didn't have a lot of supporting information in the report, and I agreed with you.
Q. Apart from those two instances, do you stand by everything else in the report?
A. Yes.
Q. If you could go to Page 2 of 38 of your report, please. You have a section that's entitled, "Data and Other Information Considered," correct?
A. That's right.
Q. And in that paragraph you wrote, "Other significant case files used in the preparation of this report include the plaintiff's Fourth Amended Complaint, the Nipper postings from December 23rd and December 24th, 2000,

correct?
A. Right.
Q. And that was based on information provided to you?
A. That's correct.
Q. If you could go to Page 18 of 38 of your report.
And you wrote, "There are two Internet postings of particular interest. One of them was posted by xbr21 on December 23rd, 2000, signed NipperClause00. I will refer to this posting as Nipper1." Is that correct?
A. Yes.
Q. And that again was information provided to you by the attorneys representing plaintiffs?
A. That's right.
Q. And based on all the information you received from the lawyers, you understood one of the key issues in dispute was whether NDS was responsible for the December 23rd posting of the so-called recipe to dump the EEPROM, correct?
A. That's right.
Q. And in your report you say that "Prior to December 23rd, 2000, those features were undocumented in the DISH Network hacker community," correct?
A. Where? I lost you. Where are you reading from?
Q. If you look at Page 6 of your report?
A. Page 6?
Q. Correct, sir.

A. Okay. Where on Page 6?
Q. Number 1, I believe it is.
A. Can you please repeat the question.
Q. Sure. In your report there, you say, "Prior to December 23rd, 2000, the buffer overflow vulnerability, the RAM ghost effect, the index variable --
A. The location and purpose of the index variable --
Q. Right. It's --
THE COURT: Just a moment. You two are speaking over the top of each other.
THE WITNESS: I would say.
THE COURT: Stop.
THE WITNESS: Got it.
THE COURT: Thank you.
Counsel.
BY MR. STONE:
Q. Looking at that section, it says, "Prior to December 23rd, 2000" -- I won't read it all in, but the components you discuss, correct --
A. Right. They're there.
Q. -- were undocumented in the DISH Network hacker community, correct?
A. Correct.
Q. And to this day you have not changed that in your report, correct?

A. Right. I haven't changed that opinion.
Q. Like you did your diagrams, which you did change, correct?
A. I did change the diagram, and we submitted the new one.
Q. And nowhere in your report is there any reference to a December 21st, 2000 posting; isn't that correct?
A. That's right.
Q. And no one ever showed you any December 21st, 2000, posting; is that correct?
A. I believe I was told that the December 23rd posting was a reposting of something that was posted on December 21st, but --
Q. Did anyone ever show you a December 21st, 2000, posting?
A. I don't think so.
MR. STONE: Thank you.
No further questions at this time.
THE COURT: Redirect.
MR. HAGAN: Thank you, Your Honor.

REDIRECT EXAMINATION

BY MR. HAGAN:
Q. Dr. Rubin, you reference something in your testimony with Mr. Stone called a resource-constraint device, but he didn't give you an opportunity to explain that to the jury.
Can you do that?

A. Sure. And yesterday Christophe Nicolas talked quite a bit about this. A Smart Card is a little computer, and it doesn't have as much memory as a computer, a regular computer. It doesn't have as fast a processor, and it doesn't have as big an address space. It's -- we call those things resources, and we would say that this is a resource-constrained device where everything really matters.

When the program is executing, the chip goes off of a clock, and every time the clock ticks, the chip can do something in its processing. And every time the clock ticks, we call it a clock cycle.

When I was looking at the code, I noticed that there were numbers in parentheses and comments along a lot of the lines of code, in particular as relating to the I/O buffer. I had no way of knowing what they were. When I met Christophe Nicolas, he explained to me what those were. Those were a count of how many clock cycles each instruction required. And I thought to myself, "Well, that's really unusual. When would you ever care -- I mean, clock cycles are milliseconds. They're very, very fast. Why would it matter that this instruction takes three clock cycles and this one takes five?" It's not a thing that you usually would care.

And he explained to me the importance in the Smart Card of keeping track of the clock cycles, because when you, for example, press a button on your remote, you want the channel to change right away. And in the I/O buffer, you're in the critical portion where bytes are being copied from the set-top box into the Smart Card, and if you're still processing one of the bytes, the next byte's gonna come in, and you won't be ready for it. And so that entire card was designed under very, very tight constraints. In fact, the comment in the code said that there were only six clock cycles left to spare.

Now, that is code that executes in a loop. Every single time a character is read in, that code executes again. So if you were to add some instructions to that, you would be increasing the clock cycles, and you would be increasing it times the number of times that you would loop around the loop.

And so I have no doubt that once they were aware of the problem, they wanted to patch it. But it wasn't a simple matter of just adding a couple of lines of code, because that could have really messed up the card, and they were very sensitive to not doing that.

And now I understand from Christophe Nicolas that when they finally did issue a patch, they completely restructured the way that they did the input and output, and that's where they ended up with this thing of checking it twice. It had to do with the new structure that they had completely rewritten, and that's why it took them a while before they could release the patch.

Q. Thank you, Dr. Rubin.

A. No problem.

Q. So is it your testimony based on what Mr. Nicolas testified to yesterday that the plaintiffs acted reasonably to develop and launch that software patch?

A. I think they did.

Q. Now, Mr. Stone also referenced to you an opinion -- that I think their expert may have -- that somehow this was an inside job, that EchoStar and NagraStar and NagraCard hacked their own system, cost themselves their own millions of dollars. Do you agree with that opinion?

A. No, I don't. I don't understand how you could think that.

Q. And can you explain to the ladies and gentlemen of the jury why?

A. Sure. If I'm setting out to build a card, and my goal is to later be able to hack the card for some reason -- I don't know if you're familiar with the term Rube Goldberg. It's a term for a crazy -- some of you are nodding, but I'm going to explain for those who aren't -- a crazy, wild contraption.

Let's say your goal is to close a hot dog. You put the bun on top, and you build something where marbles fall, and it hits a pad and launches a spring, and water shoots down. The whole convoluted thing is called a Rube Goldberg, and I view this hack as a Rube Goldberg hack. It's complicated. It's got a lot of moving parts.

If I was going to build a Smart Card that I was going to hack later in this particular system, it would be trivial, because there's something called the "entitlement management message." This is something that's sent to the Smart Card and can include code that can run on the Smart Card when it gets there. And the only thing you need to do to get that to run is know the secret key. If I'm the developer of the card, I made the secret key, I know what it is. All I would do is hang onto that key and let the card go into the field, and I would hack it whenever I wanted to. I wouldn't build something that required a buffer overflow into a ghost alias memory effect into an address index variable into a checksum. I mean, that to me is completely ridiculous.

Q. And Mr. Stone, I think, tried to make the point that the engineers for NagraCard intentionally did not check the communications I/O buffer in the ROM 3 card. Is that correct?

A. That's correct.

Q. And Dr. Rubin, it's true, based on Mr. Nicolas' testimony and your review of the source code, that they did

not check that buffer, correct?
A. That's right.
Q. And can you explain to the ladies and gentlemen of jury what you believe the explanation for that is?
A. So that's what I did a few minutes ago. Oh, I'm sorry. It's not. It's a new question. I'm getting confused.
So the reason from -- based on the comments and on the testimony that I heard yesterday that I believe that they didn't check the I/O buffer overflow is that they didn't know about the ghost aliasing effect that the memory would wrap around. And they were very, very concerned about clock cycles. And it was their belief that the code -- anything that would be written past the buffer would just dissolve into nowhere because there is no memory over there. And so to save time and to avoid using up these precious clock cycles, they decided not to check the buffer bounds there.
Q. In essence it would have been, at least in their mind, superfluous?
A. It would have been, in their minds.
Q. And while I understand that the defendants and their counsel now disagree with that opinion, David Mordinson does agree with your opinion. In fact, he has that same opinion in the Headend Report itself, doesn't he?
A. That's right.
Q. And you quote from that in your report where

Mr. Mordinson says, "The application designers did not check the maximal possible length of an incoming message while it is being collected as to determine if it exceeds the buffer. Probably they believed such a verification was superfluous. Indeed, there is no physical memory allocated from the location Ox200 to Ox1FFF.

"However, due to the RAM ghost effect, an incoming message of maximal length, i.e, 255 bytes, will affect RAM locations from 0X19C to 0X1DD and from 0X20 to 0X9A."

Do you agree with Mr. Mordinson's analysis back when he wrote the Haifa report?

A. That's right. I think that he conjectured as I'm saying now that they just didn't think about any possible damage that could happen from writing into areas of memory that they didn't think existed.

Q. Now, Mr. Stone also showed you Exhibit 809. Do you have that in front of you still?

A. Yes.

Q. And the date on 809 is May of 1999. That's when that was released, that article, correct?

A. Right.

Q. And that was coauthored by a former pirate named Oliver Kommerling, correct?

A. That's right.

Q. And you understood in the testimony of Mordinson and

Shkedy that Oliver Kommerling was employed in some respect with the defendants at the time that they created the hack for EchoStar's system, correct?
A. I honestly don't remember that. Sorry.
Q. Not a problem.
You would agree, Dr. Rubin, that you haven't seen any evidence in this case, either from Mr. Stone at deposition or from the documents that you reviewed, that suggest in any way that EchoStar's security system was compromised or hacked at any point in time prior to Mr. Mordinson and Mr. Shkedy developing a hack for that system?
A. That's correct.
Q. Now, you were also present yesterday when Mr. Nicolas testified that if Chris Tarnovsky e-mailed portions of the ST system ROM to Jan Saggiori in March of 1999, the only way he could have gotten that portion of EchoStar's code was if he either participated in the defendant's Headend Report and project or he got that information from them.
Do you have any reason to disagree with Mr. Nicolas' testimony on that point?
A. No.
Q. Now, it's not disputed, and you heard Mr. Nicolas testify yesterday, that NagraCard developed a software patch which they hoped would fix this hack methodology in early 2001, correct?

A. That's right.
Q. Now, would that software patch have been effective if the pirates were using a device called a blocker?
A. No.
Q. And can you explain to the jury why that is?
A. A blocker is something that is specifically designed by pirates to prevent a patch that's going to fix a vulnerability from actually fixing the vulnerability. It blocks it. That's why it's called a blocker.
Q. And would that software patch be effective on pirated ROM 3 cards that were reprogrammed using the Nipper or Haifa recipe prior to early 2001?
A. It depends.
This -- if -- the pirates could have built a mechanism into cards that were pirated already that would prevent future hacks from being effective.
Q. And Mr. Nicolas testified yesterday that that was the case.
Do you have any reason to disagree with that testimony?
A. No.
(Live reporter switch.)
(Further proceedings reported by Jane Rule in Volume IV.)

-oOo-

-oOo-

CERTIFICATE

I hereby certify that pursuant to Section 753, Title 28, United States Code, the foregoing is a true and correct transcript of the stenographically reported proceedings held in the above-entitled matter and that the transcript page format is in conformance with the regulations of the Judicial Conference of the United States.

Date: April 17, 2008

____

DEBBIE GALE, U.S. COURT REPORTER

CSR NO. 9472, RPR