UNITED STATES DISTRICT COURT

CENTRAL DISTRICT OF CALIFORNIA

HONORABLE DAVID O. CARTER, JUDGE PRESIDING

- - - - - - -

ECHOSTAR SATELLITE CORPORATION, et al.,	)	
Plaintiffs,	)	
vs.	)	No. SACV 03-0950-DOC
NDS GROUP PLC, et al.,	)	
	)	Day 12, Volume III
Defendants.	)	

_____)

REPORTER'S TRANSCRIPT OF PROCEEDINGS

Jury Trial

Santa Ana, California

Tuesday, April 29, 2008

Jane C.S. Rule, CSR 9316
Federal Official Court Reporter
United States District Court
411 West 4th Street, Room 1-053
Santa Ana, California 92701
(714) 558-7755

08-04-29 EchoStarD12V3

APPEARANCES OF COUNSEL:

FOR PLAINTIFFS ECHOSTAR SATELLITE CORPORATION, et al.:

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

FOR DEFENDANTS NDS GROUP PLC, et al.:

O'MELVENY & MYERS, LLP
Attorneys at Law
BY: DARIN W. SNYDER
NATHANIEL L. DILGER
DAVID R. EBERHART
MICHAEL O'DONNELL
Attorneys at Law
Embarcadero Center West
275 Battery Street
Suite 2600
San Francisco, California 94111-3305
(415) 984-8700

- AND -

HOGAN & HARTSON, LLP
Attorneys at Law
BY: RICHARD L. STONE
KENNETH D. KLEIN
Attorneys at Law
275 Battery Street
Suite 2600
San Francisco, California 94111-3305
(415) 984-8700

I N D E X

EXAMINATION

SANTA ANA, CALIFORNIA, TUESDAY, APRIL 29, 2008

DAY 12 -- VOLUME III

(1:00 p.m.)

(The following proceedings is taken in the presence of the jury.)

THE COURT: Okay. We are back in session. The jury is present. The witness is present.

Counsel, thank you for your courtesy. If you would please be seated.

This is Mr. Stone's continuing direct examination of Mr. Jones.

MR. STONE: Thank you, your Honor.

NIGEL JONES, DEFENDANTS' WITNESS, RESUMED

DIRECT EXAMINATION (Continued.)

BY MR. STONE:

Q Okay. Mr. Jones, to recap a little bit, we discussed earlier that for any buffer overflow attack on this card, there would be no choice as to memory aliasing, correct?

A Correct.

Q No choice as to using the index variable?

A Correct.

Q And no choice as to using an exception handling?

A Correct.

Q Now, where there were -- strike that.

Were there times when the author of the Headend Report

and the author of the Nipper code had choices to make?
A Yes, they had multiple choices to make.
Q And what happened when there was a choice?
A In every case, they made a different choice.
Q And what's the significance of that?
A Well, it seems to me that if two people have a choice, and they always make a different choice, then that is symptomatic of independent development, two people working on this independently.
Q Now, we've talked about some of these choices and differences before. Let's talk a little bit about the 7381 jump.
A Yes.
Q I know we've touched upon that. Why is that important?
A Okay. To reiterate, if you remember, in the Nipper code, he terminates this program by jumping to location 7381, and that jump to that subroutine required a parameter to be passed to it. The critical thing about this is that that location was not documented in the Headend Report. I can actually go further than that.

I have looked at the disassemblies that David Mordinson did. David Mordinson attached no importance to that location at all, it was just, yeah, that's some sort of interesting location. I've gone through 800 files produced by Haifa, and in none of those can I find any reference of

the importance of location 7381.

Q So how would the person who wrote the Nipper code have determined that?

A As far as I can see, there is only one way. He had the ROM, he disassembled it, he studied it, and he worked out the importance and the utility of location 7381.

Q And if the author of the Nipper code had the ROM contents and could figure that out, would he or she need the Headend Report?

A No, obviously not, no.

Q Now, in addition to the choices that were made differently in the 7381 jump, is there other evidence that Nipper code was independently created from the Headend Report?

A Yes, there is.

Q And what is that?

A What I'd like to do now is talk about how David Mordinson's code is architecturally superior.

I have a graphic up here, and I have three things that I'd like to draw your attention to.

The first one is "program download time." So this is literally, "How long does it take to transmit the program from the reader/writer to the card?" The transmission speed is fixed by the design of the card, and so the shorter the program, the faster it downloads.

If you can see, David Mordinson's system required downloading about 168 bytes. The xbr21 Nipper code required you to download 196. Now, that's about 16.7 percent slower.

Now, here is the important thing that I took from this. David Mordinson put together what we call proof of concept code. He came up with an architecture. And we then have the Nipper code, which presumably, it's a couple years later, they've had time to improve it, and so on, but it takes 16.7 percent longer to download. That's a significant deterioration in performance, particularly if what you're interested in doing is programming Smart Cards quickly. You would like your program to download as fast as possible.

Q And what is the next point?

A The next one, the maximum size of shell code, or the virus. Because David Mordinson put his program into the communications buffer, his program could be as big as the communications buffer, which meant it could be a hundred bytes long. By contrast, the Nipper code, by putting the program into the stack, he had to squeeze all his program in between the top of the index variable and the bottom of the stack -- or the bottom of the top of the stack, if you can follow that, okay?

He gets 62 bytes, so the maximum size program that the

Nipper code can download is 38 percent smaller. That means David Mordinson's approach allows you to do more per download, okay? And when I got to examine the black box, which we'll be talking about shortly, the black box went through a whole series of programming steps to achieve what they were trying to do. If they'd used the Mordinson method, they would have had to have used less downloads. They could have done it faster.

Q Now, what about the index variable?

A The index variable, we've talked about that this morning. I hope you've got a good idea of what it does.

Now, the nice thing about David Mordinson's approach is because his program always goes into the communications buffer, the IO index variable never has to change. It always says "point to there." As you change the virus or the program, you don't have to change the value of the index variable.

With the Nipper code, by contrast, the way it's structured, every time you change the program, so for example, let's say that you wanted to change the zip code in your card such that you could watch TV from New York, that was being sent to New York. Living here in Los Angeles area, you can't see the New York transmissions, but they're here. So if you want to do that, you could write a -- a program, a virus that downloads to the card and says "change

my zip code so that I can now watch New York TV." Well, with the Nipper methodology, you would then have to go and recompute the value of the index variable.

Q So is the Nipper architecture an improvement on the architecture that Mr. Mordinson developed in the Headend Report?

A Absolutely not. David Mordinson's architecture is faster, gives you a bigger payload and is easier to use --

Q So --

A -- compared to Nipper.

Q I'm sorry.

So if the explanation by plaintiffs for the differences between the Nipper code and the Headend Report is that it took years to improve the Headend Report, does this evidence disprove that explanation?

A In my opinion, yes.

Q Can you tell us the reasons you believe the Nipper code was independently created?

A Yes, I think so.

If we look at sort of the three basic things, okay? The first is I've shown you all these differences between the Nipper code and the Haifa code, okay? I listed 10; I said there were more, all right?

The second thing is what we've just covered here, and that is -- excuse me -- that the Mordinson code is superior.

And the last thing is, I hope I've shown you that the four pillars that Dr. Rubin identified as being indicative of any -- indicative that any tactic uses those must have come from Haifa is wrong, because any buffer overflow attack developed independently by anybody must use those.

Q So have we covered all of your bases for your first opinion that the Headend Report was not the source of the Nipper code?

A Yes, sir.

Q Now, I'd like to go to the second internet posting that we've talked about in this case, and if we could put up Exhibit 191, which is in evidence.

So we had the xbr21 posting on December 23rd of the so-called recipe, and now we have this posting by somebody using the nickname NiPpEr2000 on December 24th. Have you had a chance to analyze this posting?

A Yes, I have.

Q And briefly, what is the difference between that posting and the xbr21 posting?

A Yes. The xbr21 posting was, if you like, a -- a program that allows you to download the contents of the EEPROM. That was that one book I showed you in the animation. What the NiPpEr2000 is, is the contents of that EEPROM. That's literally the book, okay?

Q And is the EEPROM the same for all NagraVision access

cards?

A It's similar, and so in the EEPROM, they put things that are asked specific to a particular user, like the card ID and what channels you are allowed to watch, and so on. So when you look at them, you'll see a lot of similarities from card to card, but some differences.

Q Now, was the NiPpEr2000 posting the first time that EEPROM contents of the NagraVision card had been posted on the internet?

A No. It was the third occasion.

Q What were the prior two occasions?

A Yes. In 1998, an EEPROM image from a NagraVision card was dumped by someone called "Swiss Cheese Productions."

Almost exactly a year later in October of 1999, a second EEPROM image appeared, this time by someone using the handle "Fruitcake," and then we have the NiPpEr2000 posting.

Q And I'll come back to the Swiss Cheese and the Fruitcake postings in a little bit.

Focusing on the December 24th, 2000 posting, you understand the allegation is that NDS or its employee were the source of that posting?

A Yes, I do.

Q And have you examined that?

A Yes, I have.

Q And do you agree with that allegation?

A No, I do not.

Q And do you have a chart showing the five reasons why?

A Yes, I think so.

Okay. The first thing I'd like to tell you is the format of the NiPpEr2000 posting. We just had it up there on the exhibit. If you'd looked carefully, you would have seen each line starting with a colon and then a 1-0, and then a whole bunch of, essentially, gobbledegook. That format is what's called an Intel HEX record file format, something that engineers use for transporting memory images around.

The interesting thing is, is that I examined about 800 files produced by Haifa, a huge number of which contained EEPROM images in one form or another, and none of them were in that format.

Q What about the second point?

A So the second point here, the information on Nipper's card was very different than the information on Haifa's card. What I mean by that is, is I previously told you that you can personalize the cards, so some of the things that get stored on the card include things like the last pay-per-view you watched, when was the card activated, when was the last time you watched TV, okay? And so by looking at that information, I can see that the NiPpEr2000 card had been in use well before Haifa started their reverse

engineering, and it was still actively in use in 2000.

Q Now, the third point says "None of the NagraStar access cards from which Haifa extracted EEPROM contents was the source of the Nipper posting"; how do you know that?

A Yes. I think I have a slide that shows this.

What I'd like to show you here, ladies and gentlemen, is you can see that I've extracted something in big green, okay? What this is, is the so-called "backdoor password." This is unique to each card, and this is -- if you know this password, you can get in and obtain complete control of the card. So when this image was posted on the internet, this backdoor password was not removed, okay, which meant that NagraVision could take that password and say "Whose card does this belong to?"

Q And are you aware that that has been done?

A I am aware that it was done quite recently at our request, yes.

Q And are you aware that it has no connection to NDS?

A That's my understanding, yes.

Q Now, let me show you -- keeping that backdoor password in mind, but let me show you Exhibit 206.

And this is in evidence. If you could zoom in to the language that says "Charlie, please fry these cards."

BY MR. STONE:

Q Now, having become aware of this, you need the

password. Does the language in this posting that says "Charlie, please fry these cards. We don't know how to unloop, yet," make more sense to you?

A Yes, it does. The first thing I need to tell you is that Charlie is what the pirates refer to as EchoStar, because Charlie Ergen is the CEO. So when you say "Charlie," they are referring to EchoStar, okay? So this means "EchoStar, please fry these cards."

Q And how would that work?

A Well, here is the interesting thing. If what you want to do is to disable a card that you think is a pirate card, then you have to have some criteria for identifying the cards you wish to fry. Well, this posting, by putting in there this unique backdoor password, is giving NagraVision a huge hint. "Any card that is being cloned from this one that's got this backdoor password in, kill it." That's my interpretation of what this posting means.

Q Would that assist or hurt EchoStar?

A Well, I can't see how it would hurt them, so I assume it must assist them.

Q And what would happen if the image on the internet were simply downloaded by somebody into their card?

A Quite frankly, disaster. What would happen is the way the EchoStar system works is that when you subscribe to EchoStar, the card and the receiver get what they call

"married." In other words, it's now a matched set. If you downloaded this into your Smart Card, you would no longer be married to your receiver, would not be able to get TV. And I might add, that's the best that would happen to you. There's a few other things that could happen, which are even worse, as in completely destroy your card.

Q Now, you mentioned the -- the word "cloning." Is that the same thing as copying an image of a card?

A Yes.

Q So if anyone copied the image of that card, it would be ripe for an electronic countermeasure based on the backdoor password?

A Correct.

Q And if it wasn't married to the correct receiver, it would also crash the card?

A Correct.

Q Now, let me go to your next opinion, which is that a hack of this system was inevitable.

And we talked earlier about some EEPROM dumps that preceded the one on December 24th, 2000, I believe. So the first one was by Swiss Cheese Productions --

A Yes.

Q -- in 1998, and then Fruitcake in 1999?

A Correct.

Q And have you formed an opinion based on the evidence

whether the evidence shows that those EEPROM dumps were derived from Haifa or the Headend Report?

A Yes, I have.

Q And what's your opinion?

A I've examined the images, and based on my analysis of the images, I can find no evidence that links them to Haifa.

Q Okay. And what did that consist of?

A What I did is I looked at the images and said, "Is there any similarities here between what is published and what is it that Haifa has in their possession?" And so in both the Swiss Cheese Production image and the Fruitcake image, they sort of added comments and said things like "verify keys here," and so on, so to try to explain what various things were in their posting.

Well, the interesting thing about that was that the terminology they used was different than the terminology used by Haifa, but it was more than that. Fruitcake and Swiss Cheese Productions couldn't even agree to call the same thing the same thing. And it got even worse than that, because they called the different thing, something that was different, the same thing. So you've got all this mishmash of terminology between Fruitcake and Swiss Cheese Productions, and none of that terminology matches the terminology used by Haifa.

So the second thing I did with the Swiss Cheese

Production image was said to myself, "Well, can I compare what was put on the internet to anything that Haifa has?" And so I identified all the possible EEPROM images in Haifa's possession that could possibly have been the Swiss Cheese Productions image, and I did that based on what's called the revision string number and also on a decryption key.

Q And do you have a chart of that?

A Yes. So I appreciate this is a little difficult to read, but on the left-hand side, these are the lists of all the EEPROM images from Haifa that had the same revision number and the same video key as Swiss Cheese Productions. So what I did is within these files, I redacted the information in them in exactly the same way information in Swiss Cheese Productions' posting was redacted. In other words, I was going to compare apples to apples. Then wrote a little computer program and said, "Please compare these files." And this shows that in every case, there was a difference. Actually, in every case, there were multiple differences. What I'm showing you here is where the first difference occurred. And so it was clear to me that the Swiss Cheese Productions image did not come from any card in Haifa's possession.

Q What about the Fruitcake EEPROM dump that was posted in October of 1999?

A Right, this one was fascinating, because one of the fields that's in the EEPROM image is what's called a network ID, and the network ID identifies where that card comes from. The Fruitcake network ID indicated that that card came from Spain. See, NagraVision deploys their system all around the world, and so whomever published that EEPROM image had a Spanish system, so someone in Spain, essentially.

Q Now, going back to the Swiss Cheese posting, did that have the same video key as the cards that you examined from Haifa?

A Yes, it did.

Q And did that change your opinion?

A Ultimately, no.

Q And why not?

A Well, I did several weeks' analysis looking at these video keys trying to understand them, and I ultimately came to the conclusion that the video key that was in both images was the default video key. So when you make a Smart Card, an access card for satellite TV, you have to put something into the EEPROM, and so NagraVision loads in a default video key.

So I did my analysis, and I came to that conclusion, and then a couple weeks before the trial started, I finally got to see the source code. And when you go into the source

code directories for the ROM 2, the ROM 3, the ROM 10 card, they have a directory that says "databases." You go into that, and in there, they contain the default EEPROM image that gets loaded on the card. And surprise, surprise, the video key that's in those files matches the one in Swiss Cheese Productions, so it's just the default video key.

Q And that would be in every card?

A Every card, yep.

Q Let's go to -- back to StuntGuy for a moment. You testified that StuntGuy's claim to fame was this "Frequently Asked Questions" document?

A Yes.

Q And was that available on the internet?

A Yes.

Q And was that available beginning in 1999?

A Yes.

Q What was your reaction when you first saw the StuntGuy "Frequently Asked Questions" posting?

A Quite frankly, I was stunned. I -- I looked at the information in it, and I said, "There is no way this guy is working this all out by himself. He has to have inside information."

Q By that, you mean access to secret documents from EchoStar or NagraVision?

A Exactly.

Q Did you see any evidence that StuntGuy had access to such documents?

A Yes, I have a slide here that shows that.

Q And what is this?

A What this is, this is an excerpt from the StuntGuy FAQ, frequently asked questions, and he produced two versions of these, one for public consumption and one for just him and his close buddies. And this was called the unabridged version. And you can see here it says "The additional information as to data type, field sizes and meanings, commands, and so forth, come straight from the EchoStar receiver source code."

What that means is StuntGuy had access to the programs that EchoStar was writing and putting in their receivers. I mean, this is a first order magnitude security breach. Your source code is the most important thing that you keep secret.

And what happened here was that EchoStar was putting this up on what's called an FTP site. FTP stands for file transfer protocol. When you download stuff off the internet, most of the time, even though you may not know it, you're downloading from an FTP site. And so this information was on an FTP site that one of StuntGuy's associates got into and extracted the code --

Q And --

A -- and gave it to StuntGuy.

Q I'm sorry.

Did you also set out to determine whether there was any connection between StuntGuy and the Headend Report or NDS?

A Yes. So an obvious question would be, StuntGuy has produced this hundred page document detailing a tremendous amount of technical information, did he develop it independently, or did he get it from Haifa? Well, what I discovered was that Haifa knew things he didn't, he knew things they didn't, and there was a few things in which they flat-out disagreed on, okay? So it was pretty clear to me that StuntGuy was not getting his information from the Headend Report.

Furthermore, StuntGuy was identified, and you're going to be hearing from him in a couple of days. And when he was deposed, he was flat-out asked, "Did you get that information from NDS," and I believe he answered "no."

Q So have you told the jury all of your reasons why you have concluded that NDS is not responsible for the December 2000 postings, the StuntGuy, FAQ, or any of the other information on the internet, including the Swiss Cheese posting?

A Yes.

Q Now, let's talk about the NagraVision system. That remains secure for about three to four years, correct?

A Yes, it did.

Q And doesn't that mean that it was a better system than the NDS system?

A No, that's not the case. Let me explain.

Back in the mid-1990s, EchoStar was actually the third satellite TV company. There was DirecTV, PrimeStar, and then EchoStar. DirecTV had been out for a long time -- much longer time. They had by far and away the most subscribers, and they also had the best programming. So if you're a pirate looking to hack a satellite TV system, which one would you go after, a company with millions and millions of subscribers, great programming, or a company with very few subscribers and inferior programming? I'd go for the big fish, and that's exactly what the pirates did.

Q And do you have a chart that illustrates this?

A Yes. What this shows you is the dates that NagraVision introduced their various cards and when they were hacked. As you can see, the ROM 2, 3 card, which is very similar to each other, was introduced in 1996, and it was hacked late 1999.

The ROM 10 was introduced in 2000, and it was hacked in 2002, so we've gone from three years down to two years. The ROM 11, to the best of my knowledge, was introduced in 2001, and it, too, was hacked in 2002, so it took a year.

And then we have Aladdin. Now, just so you understand,

you've heard testimony about card swaps, okay? The card swap is they are going to swap out ROM 2, 3 and 10 and 11 and replace it with the Aladdin system, NagraVision's latest system. As you can see, Aladdin was hacked on arrival. This is their latest and greatest, which presumably is much more secure than the system they are replacing.

Q Now, is it your understanding that around 2002 is when the DirecTV system became more secure with the P4 card?

A Yes.

Q And what influence did that have on this chart?

A Well, DirecTV had a whole series of -- of cards supplied by NDS called P1, P2, P3 and P4. Each successive generation got tougher to crack, and -- but the pirates felt very confident that the P4 would be hacked, just as the previous generations had been. Well, it turns out it hasn't been. Here we are quite a few years later, and the P4 card is basically secure. And so the pirates are looking at this saying, "Oh, oh, we can't get past the P4 security, so we need to turn our attention elsewhere," and they turn their attention to EchoStar.

Q Now, how good was the security in the EchoStar receiver?

A Well, actually, embarrassingly bad.

Q Why do you say that?

A Well, I have a slide here.

Actually, before I draw your attention, remember we did the animation, and I showed you that lockbox where the decryption key gets put in the lockbox and it goes over to the receiver, okay? The receiver unlocks the box and extracts the encryption key. Well, that's very close to what actually happens, because the message that is sent over to the receiver is unlocked using what is called the "box key." That's literally what the pirates called it, okay?

As you can see, this box key is pretty important to the security of this system. Well, it turns out that you could get the box key merely by pressing some keys on your remote control. So this is an excerpt from the Dover FAQ.

Q Was this posted on the internet?

A Oh, yes, very early. And he tells you, "This is how you dump the memory. Press menu, select 6, select 3, press info, right arrow, left arrow," and you've got a dump of the memory of the receiver.

There is another page form this that basically says "Once you've got this, if your receiver is this model, go down to here, and there is your box key." I mean, to me, that was amazing.

Q And this was all on the internet?

A All on the internet, yes.

Q Now, have you heard of a device called a JTAG?

A Yes.

Q What is that?

A Well, as you can imagine, what EchoStar did when they realized that this is how people were getting their box keys, they reprogrammed all their receivers to stop this happening. Well, that turned out to not put even a dent in the road, because the design of the receiver is fundamentally flawed in that these box keys reside in external memory. And to read external memory, all you need is what's called a JTAG device, okay.

Q And what is that?

A JTAG stands for Joint Test Action Group, and it's a standard by which you can interrogate chips, okay?

Here is a website you see compatible with all receivers that have a JTAG port, price, $39. So for 39 bucks and a piece of pirate software, you can get your box keys like that.

Q Now, did NagraVision make some mistakes regarding the security of their system, in your opinion?

A Yes, they did.

Q And what was -- or what are those mistakes?

A Well, the first -- in my opinion, the first big mistake they made was in their choice of microprocessor, okay? And the microprocessor they chose, in my opinion, had three fundamental problems.

The first is it had what's called a Von Neumann

architecture. Now, computers can be either a Von Neumann or a Harvard architecture. And I won't go into the details of them, but suffice to say, if this had been a Harvard architecture microprocessor, this attack would not have been possible.

The second thing, of course, is they chose a processor that aliased memory, okay? Now, NagraVision can say, "Well, we didn't know that, it wasn't supposed to," and so on. But the first tenet of building a secure system is validate your hardware platform, okay? Make sure that the thing you're using does everything it's supposed to do in the way you expect. The -- the onus is upon you as the user, and --

Q And what is the final point?

A The final point here is that the process that they chose lacked sufficient processing power. And the consequence of that is, is that most of the messages that flowed between the receiver and the Smart Card went unencrypted, because encryption took too long. And the pirates, as a result, were able to learn a tremendous amount about this system, because all the messages were going back and forth literally in the clear.

Q And what was the second major mistake, in your opinion?

A The -- excuse me. The second major mistake, by far, was the failure to check for buffer overflow.

MR. STONE: And your Honor, may I approach?

THE COURT: You may.

MR. STONE: I think we may have a chart of that.

For the record, this is 1586, which is a blowup of some comments to the original ROM 3 card.

THE WITNESS: Okay, ladies and gentlemen, what I'd like to show you here, this is an extract from the NagraVision code that I got to see a few weeks ago. Now, when I first started working on this, I knew that they weren't checking for buffer overflow, okay? I didn't know why. This tells me why.

So the author is saying, "Note, that one should better check the value of APDU Index" --

(Interruption in the proceedings.)

THE WITNESS: "Note, that one should better check the value of APDU Index to make sure it does not go on beyond the end of IO buff."

So the person that wrote this is saying, "Yes, absolutely, I know I should be checking this, but since there is no memory there, we skip this to save time."

Very interesting comment. "I'm not going to make a critical check, because there is no memory there, to save time."

BY MR. STONE:

Q Have you had an opportunity to analyze the two reasons given by the author of the code for not checking the buffer

for the communications buffer in the Smart Card?

A Yes, I have.

Q And to be clear, that buffer is the one buffer that can be used for a malicious attack on the card, correct?

A Yes, the one and only.

Q Now, the first reason was saving time. Did you find that explanation to be credible?

A No, I didn't, and here is why.

Could I have some video, please.

MR. STONE: The slide.

THE WITNESS: Thank you.

So here is the interesting thing. I think you heard -- I think it was Mr. Nicolas talk a little bit about this, and he explained that there were only six clock cycles left, and so Mr. Osen was very keen to save time. Well, how many clock cycles does it take to perform the check? The answer is five. So there is enough time to perform a check.

The second thing is, this check is performed after a character has been received, and is done in what's called the guard time. So, in other words, when characters come in, okay, if you like, you can put a little gap between them, and that's called a guard time. And the purpose of the guard time is to allow you to do things like process the information you've just received, okay?

The specification that these cards are written to

actually recognizes this is a problem and allows the Smart Card to say to the receiver, "I need a guard time of," whatever, okay?

Now, NagraVision chose to tell the receiver, "We don't need any guard time at all. You can send us the bytes back-to-back with no gap."

And so the final thing you need to know is that as of when they did get around to patching the card, they changed the architecture slightly of the way they handled the communications, and by changing that architecture to close the hole, they actually had enough time to check it, not once, but twice, okay? And so I put all those together, and I find the assertion that there was insufficient time as being a reason for not checking to be very strange.

BY MR. STONE:

Q Now, Mr. Jones, when I questioned Mr. Nicolas, he made a pretty funny joke at my expense that doing this would cause the remote control to slowly pull up the channels; is there any truth to that?

A I don't believe so.

Q Why not?

A Remember when I showed you that animation of the message coming in from HBO, okay? Then the time to change channels consists of the time to send the message to the card, the time for the card to process the message, the time

for the card to encrypt it, that's put it into that lockbox, and then transmit it back, okay? All those various steps.

If NagraVision had increased the guard time by the smallest amount possible, one-bit time, they would have had had more than enough time to do the checking, and here's -- here's the rub. It would have slowed down the communications from the receiver to the card by 10 percent. Everything else would have been completely unaffected. But here is the thing, communication from the receiver to the card is the fastest thing -- well, the slowest thing in it, by far, is the encryption, okay? So slowing down just that portion of the communications would have had an absolutely negligible impact on the speed of which you would change channels.

Q Now, the second point was that there was no memory out there. Did you analyze that?

A Well, here is the thing, okay? I've got this slide up here, "points of certainty." You've heard ad nauseam the buffer overflow attacks are the most common. Anybody involved in computer security knows about these. The author of the code says, "I'm not going to check, because there is no memory out there," but it turns out there is.

Now, to my way of thinking, if I'm going to make such a bold assertion that there is no memory out there, why didn't he check? You know, it -- it baffles me. I would have --

if I had made this assertion, I would have performed the check.

Q So did you find the reasons given by the author, Mr. Osen, for not checking the buffer -- buffer as holding up under scrutiny?

A No, I did not.

Q Now, is giving a false reason for something a classic indicator of a deliberate attack or defect?

A Yes, I think so.

Q And we'll come back to that in a second.

How easy would it have been for NagraVision to put the code in to prevent the buffer overflow?

A Trivial.

Q What was the -- the third major mistake, in your opinion?

A The third major mistake concerned the way they set up the memory access control matrix, okay? This is our little security guard that I wanted to mention to you before.

So the job of this thing called a memory access control matrix is to control who has access to what portions of the card, okay? This was something that the manufacturer of the chip built into the card explicitly to help prevent malicious attacks, okay? NagraVision chose the weakest possible setup for the memory access control matrix that they could.

Q And do you have an animation that --

A Yes, so what I'd like to know is -- I'll show you an animation of what would have happened if they'd set up the memory access control matrix in a more robust manner.

Could we run the animation, please.

So you see now we have a security guard. We have our big message coming in, okay? It goes in the IO buffer. The IO buffer comes, overflows, but there is a security guard there stopping the virus from going anywhere, and that security guard shepherds it over and throws it in the trash.

Q Is there any downside to using the memory access control matrix to do that?

A There is. The disadvantage to them doing it is it makes doing ECMs much harder. So they made a trade-off there to their detriment.

Q And was the backdoor password that allowed placing the card in system mode a mistake, in your opinion?

A Yes, this was a gigantic mistake.

Q Why do you say that?

A Well, let me just remind the jury. Remember, I had that slide a little while ago with that big, green number on it? Okay. That was the backdoor password. With a backdoor password, you can get into the card and do absolutely anything, all right?

And so these are the three things it will allow you to

do. Read almost any memory in the system, write to RAM or EEPROM, and download and execute arbitrary code. So you've got this, you've this card in its entirety.

Q And is it just your opinion that that is a mistake?

A No. Putting something like this into a card is a colossal mistake, and so what I'd like to show you here is an excerpt from Rankl's Smart Card Handbook. And just so you can all see, this is Rankl's Smart Card Handbook, okay? This is the book on Smart Cards.

This is what Mr. Rankl had to say: "It would be possible to use them," he is talking about backdoors, "to read the secret keys in real Smart Cards. In order to eliminate the possibility of such an attack, a creation of dump commands is prohibited as a matter of principle."

Q What does that mean?

A What that means is he's saying that doing this is such an incredibly bad idea that even if it slows you down and makes your life miserable, you must not do it. It's too dangerous.

Q Did you see any evidence that NagraVision knew this was a mistake?

A Yes, I did, actually, and I have this, because I've seen the ROM 3 code and the ROM 10 code. What I'd like to do now is show you a comparison between ROM 3 and ROM 10.

Q And this is from the source code that you reviewed a

few weeks ago?

A Correct.

So this is a password where you log in like you do on any computer system. So on a ROM 3, it's just a straight what we call textual comparison. You've got the password; you're in.

On a ROM 10, it was some sort of cryptographic algorithm. On a ROM 3, you could try as many times as you liked. It would never reject you, just keep on trying. ROM 10, you've got three choices -- chances, excuse me, and then you were done. The ROM 10 prohibited access to RAM, read or write.

Q And just to be clear, when was the ROM 10 code developed?

A The ROM code 10 was developed in 1999, early 1999.

Q And how do you know that?

A I've looked at the dates on the files.

And then last, but not least, you can see that access to the user ROM and downloading and executing arbitrary programs are also prohibited. So NagraVision in 1999 said, "You know what, we got it wrong on the ROM 3. We are going to dramatically curtail back its capabilities for the ROM 10 card."

Q And that was in 1999?

A 1999, yes.

Q Now, we've heard some testimony about security sensors. Do you believe that Nagra made a mistake with respect to the security sensors?

A Yes, I do.

Q And what is your opinion based on?

A Okay. In the ROM 3 card, they have something called "security sensors."

Now, you remember that glitching animation I showed you? Okay. So what happens, in reality, on a chip when you glitch it is you change the power to it, or you change what's called a clock signal to it. Well, the -- the processor that NagraVision used had in it circuitry to detect that someone was doing this, okay? But when I looked through the ROM 3 code, I could see no evidence to suggest that NagraVision was making use of them. But when I looked in the ROM 10 code, they were. And at this point, I was quite puzzled at why aren't they using it, and I found out the other day when Mr. Nicolas was testifying. And I don't know if you recollect, but I was very interested at this point.

So what he said was, "Well, yes, we tried enabling these sensors, but in our environment, they were completely unreliable, because they would trigger too often, and so we chose not to use them."

So let me start off, here. What he is saying is "We've

chosen a processor that will detect a glitching attack. We test it, we find out it's no good, but we use it anyway." I found that startling.

Q Now, were there other mistakes that caused you to be more suspicious, shall we say?

A I'm sorry, let me get my --

Q That's two slides; one more slide.

A Right. What we have here is what I call the case for an insider attack, okay? And what this picture shows you is a whole series of stop signs, and the idea is here, if NagraVision had made a different decision at each of these stop signs, this attack wouldn't have been possible. So let me just walk you through them.

We've talked about CPU architecture. We've talked about the fact of aliased memory, we've talked about the weak MACM setup.

The next one I haven't mentioned, buffer location. The communications buffer in the ROM 3 IO buffer was placed in exactly the right place to make this attack possible, which is another way of saying it was put in the wrong place, okay? We've discussed the fact that they didn't check for buffer overflow.

Exception handler, so I was talking about that this morning, okay? The fact that you can exploit this exception handler to run your virus. Well, there are multiple ways of

implementing an exception handler. In this card, the exception handler was written in just the right way to make this attack possible.

Then we come to the last one, which I call "code reviews," okay? Now, let me explain what a code review is in the industry. A code review is when you've written a computer program, you get up in front of your peers, and you have to defend it. They basically try and rip it apart, find all the problems with it, suggestions for improvement, and so on, okay?

With this card, if this card had gone through a code review, as I understand a code review to be, there is no way that comment about checking for buffer overflow would have got through that code review.

Now, here is the interesting thing. Mr. Nicolas testified that they did indeed perform code reviews. In fact, he did them. Well, let me remind you the time line. Mr. Nicolas joined NagraVision straight out of school. He goes onto a project, which is being run by a principal engineer, Karl Osen. It takes about 15 years to reach the rank of principal engineer in most organizations. Six months later, Mr. Nicolas' job is to review Mr. Osen's work. Well, I can tell you this. When I left university, I didn't know anything. Actually, I didn't know anything for about two years, okay? And the thought of someone suggesting to

me that I go and review a principal engineer's work would have been -- I can't, you know, I -- I'm out of my league.

Q Now, do you know for sure that this was an insider attack?

A No, I do not.

Q Is there a fact -- one fact, that if you knew, would cause you to have a firm conclusion?

A Yes.

Q And what is that fact?

A If I knew that Karl Osen knew this card alias to memory, then this would be an insider attack.

Q And are these -- all of these facts we've talked about the basis for your opinion that a hack of the ROM 3 was inevitable?

A Absolutely.

Q And that's based on these various defects or mistakes that we've talked about?

A Yes.

Q Now, did you see any evidence that the system was hacked?

A Oh, yes, without a doubt.

Q Let's talk a little bit about the black box. Were you able to examine the black box files?

A Right. So what we are talking about here is that in October 2000, NagraVision got access to a so-called black

box device, and when they examined the black box device, they discovered that it was indeed hacking their cards via a buffer overflow attack.

Q Was the black box available for inspection?

A Not to me, no.

Q What is it you were able to look at, if anything?

A NagraVision provided me some very poor quality screen shots or screen captures of information flowing back and forth between the black box and what I assume is a computer screen, but I can't tell.

Q Did you perform a comparison of the black box files that did exist with the Headend Report to reach a conclusion?

A Yes, I did. So I just mentioned that the black box files were using a buffer overflow attack to hack this card, and as part of that attack, they were dumping, not the entire EEPROM contents, but part of it, but certainly enough for me to compare the two methods. And so what I did is I took the -- the binary data that was flowing back and forth and converted it in exactly the same way as I did for comparing Haifa to Nipper. I went through exactly the same things, I compared the binary images, I compared the source code images. I looked at programming styles, program lengths, you know, how they generated invalid checksums, and so on, and I found absolutely no correlation between black

box methodology and the Haifa methodology.

Q And do you have a chart on that?

A Yes, I think so.

So, again, this is an exhaustive list of the differences, okay? But I'll just take you through them one by one.

So program size we've talked about. The coding sequences were completely different.

Output routine, remember, we talked about this with Nipper, that Nipper used the built-in output routine and David Mordinson didn't? Well, black box also used the built-in routine.

Program termination. David Mordinson put it into an infinite loop. Black box jumped to location 7848. They generated an invalid checksum with different values, and so on, through it.

Q Let's talk about the termination of the program. Do you have a slide on that?

A Yes, I think so.

So you've seen something like this before. On the bottom, we have Mordinson's code, the "BRA Dollar." Remember, that's "branch always to yourself, infinite loop."

(Interruption in the proceedings.)

THE WITNESS: I'm sorry. "BRA Dollar," "branch

always to yourself, an infinite loop."

Above that, we have the black box code where they are jumping to location 7A48. Well, you've sort of heard this story before, but 7A48 does not appear in the Headend Report.

Q So what does that tell you about how the author of the black box code was able to figure that out?

A Well, evidently, if 7A48 isn't in the Headend Report, the only place they could have got it from is the ROM. But if they had the ROM, why would they need the Headend Report?

Q Now, when you say "ROM in the black box," do you mean user ROM, system ROM or both?

A Well, to do this hack, just the user ROM, but it -- it actually turns out that the black box guys seem to have the system ROM as well, and I can show you this.

This is a printout of one of the files that NagraVision supplied to me, and what's highlighted here is a JSR2179. That so happens that that location is in the system ROM. So this much I do know, is that the guys programming the black box knew enough about the system ROM to be able to call functions built into it.

Q And can a buffer overflow attack, such is described in the Headend Report, be used to obtain the system ROM?

A No, it cannot.

Q Okay. So both the black box and Nipper code used a

buffer overflow attack, correct?

A Correct.

Q Black box had the system ROM, correct?

A Correct.

Q Both Nipper and the black box used a jump routine that Mordinson did not place any significance on, correct?

A Correct.

Q And did not appear in the Headend Report, correct?

A Correct.

Q And what was the significance of that to you?

A Well, you put all those together, and for me, this shows that these two programs, Mordinson and Black Box, were independently developed.

Q And do you have a chart of the three items we've been talking about?

A Right. So what I'd like to do now, ladies and gentlemen, is put this all together for you, and I've got a nice, pretty picture.

May I have the microphone, actually? I'd like to stand up.

So what I've done here is I've put the black box code, the Nipper code, and Mordinson's code side by side, and I've color coded them. So where you see the light blue is the location of the shell code or the virus. Where you see the dark blue, this is where we are aliasing through the

registered space. The red is important, stack setup, and so on, okay?

I'll again reiterate that just because something is in the same color doesn't mean what's in there is the same. It's just that it's the same broad, general function.

Now, for me, I look at these three pictures, and I ask myself, which one is the odd man out? And it's fairly clear to me that the Mordinson architecture, the way he put his program together is significantly different to these two guys, which I think you can see have a lot of similarities.

Q Now, let's go to your third major opinion, and that's that NagraVision knew about the problem with their system and chose to do nothing about it, correct?

A Correct.

Q And have you analyzed that and prepared a time line to show that?

A Yes, I have.

So what I have here is my understanding of how things worked from a time perspective. So in 1995, this is when NagraVision first wrote the code for the card, so sometime in 1995, this is when the decision was made not to check for buffer overflow. I do know that in 1999, they wrote code for the ROM 10, and for the ROM 10, they do check for buffer overflow.

Sometime around the middle of 1999, the first rumors

start appearing about E3M cards. I understand you know what an E3M card is?

(No audible response.)

THE WITNESS: Okay, good. And by late 1999, E3M cards are starting to appear.

Now, let me put this in perspective for you. An E3M card represents a complete hack of the system, okay? If I was the CEO of NagraVision, alarm bells would have been going off all over the place. And what I would have done is I'd have got the best and the brightest in the company and said, "Work out how they are doing this, okay. How are they doing this? Tell me." I've seen no evidence to suggest that that happened.

Then we have testimony from Mr. Nicolas where he says in Q2, 2000, which I guess is spring, he has this meeting with SGS Thompson in France. And SGS Thompson informs him that this chip aliases memory.

Now, you saw the chart there where Mr. Osen says, "You better check for buffer overflow."

"But I don't, because there is no memory out there."

So as soon as SGS Thompson tells Mr. Nicolas, "Hey, there's memory out there," then Mr. Osen's statement should send alarm bells ringing all throughout the company.

Q When you say "there's memory out there," you mean

because of the process of memory aliasing?

A Exactly, okay?

So at this point, even if they don't know the exact detail of how this attack is being done, what they do know is that their card is vulnerable to a buffer overflow attack. And so at that point, they could have developed a patch to close that down, but they didn't, okay?

So what happened next? Well, in October 2000, they take possession of a black box. And that black box device shows conclusively that a buffer overflow attack is being used to hack their card and spells out the exact down to the byte representation of how it is done, and they do nothing.

We then go to December 2000 and the famous Nipper postings. The Nipper postings appear. At that point, NagraVision starts work on a patch. Seven weeks later, they have designed, developed, tested and deployed that patch, such that by the middle of February, the card is being patched.

Q Now, have you heard an explanation for why the ROM 3 cards were not patched prior to the postings?

A Yes, I have.

Q And did you agree with that explanation?

A I found it a little strange. My recollection of what Mr. Nicolas said was this. He said, "Well, if we patch the card, we will tip off other pirates to the vulnerability.

And so because we don't want to tip off those pirates, we aren't going to patch it."

Q And what is wrong with that logic?

A Well, it seems to me that if the patch patches the card, you don't have to worry about the pirates coming in.

Q Now, Mr. Nicolas' explanation was that "We didn't want to deploy the patch, because it might alert the pirates," but what about designing the patch so that it could have been used sooner?

A Right. So let's just say that Mr. Nicolas' explanation of why they don't deploy it makes sense. What doesn't make sense is, why didn't they develop it? Because the deployment is easy, you just push the button, okay? It's designing -- developing, designing, testing, that's the difficult part, the time-consuming part. They could have done that, in my opinion, Spring of 2000, as soon as SGS Thompson told him about it, okay?

But you could argue, "Well, we didn't know the exact details." Okay. What about October 2000 with the black box? Well, it took them seven weeks to do the whole process after the Nipper postings, okay? End of December to the middle of February. Presumably, if they'd started work in the middle of October when they had the black box, it would have taken them seven weeks. By my calculation, middle of October plus seven weeks takes you to the beginning of

December. So they could have had that patch developed, ready to go three, four weeks before the Nipper postings.

Q Now, is there any way that NagraVision could tell if a card had not been patched?

A Yes, so the reason this question is important is, well, what about those cards that don't get patched, okay? I think this was Mr. Nicolas' concern. Well, there is a couple of ways of telling.

The first way is, you can send down what's called an electronic countermeasure to the card, and that electronic countermeasure can essentially see, is the patch here, yes or no? The method by which it does it is typically based upon what's called a checksum analysis, but it's pretty easy to do.

There is a second method, which is even more elegant, okay? You program the receiver to perform a buffer overflow attack on the card. Why not? You can reprogram your receiver. So you say, "Okay. Perform a buffer overflow attack." If the attack is successful, the virus that you download destroys the card, because only if the card isn't patched can you download the virus.

So there is at least two ways that NagraVision could tell if a particular card had been patched or not.

Q Now, let's go to your last opinion, which is the patch that was, in fact, used, was completely effective in closing

the buffer overflow vulnerability, correct?

A Yes.

Q And can you explain what you mean by a patch.

A Yes. I'm going to show you another nice animation here, and let's just talk a little bit about terminology, okay?

When the pirates find a weakness in a card, they call it a hole, okay? A hole you can climb through. And so what do you do with a hole? You have a hole in your sock, well, you patch it, okay? And so what a patch is, is a small piece -- a small program that gets downloaded to the card to cover up the hole, the weakness.

Now, patching the -- the NagraVision card was designed explicitly to support patching, and throughout the life of the ROM 3 card, about on average, every three months a patch was issued, okay? So this is not a new procedure for them. This is something they are used to doing. And so what we're talking about with a patch, here, is a program to close the hole.

So if you go ahead and run the animation, I think this will tell the story for you.

We have something come in, plugs the hole. Now, along comes our computer virus, goes into the buffer and tries to overflow it. It can't. If it can't overflow the buffer, it can't get onto the card, simple as that.

Q Now, did you have an opportunity to analyze the actual patch code that was used for the ROM 3 card?

A Yes, I did.

MR. STONE: And your Honor, may I approach?

THE COURT: You may.

MR. STONE: And for the record, this is marked as 1587.

THE WITNESS: Okay. What we are looking at here is an excerpt from the actual patch code that I examined a few weeks ago.

And up here (indicating), it says "Interception of protocol T1 overflow," which is computer jargonese for "We're about to catch the buffer overflow."

And if you'll see the comment here, it first says: "ARRrrrrg," and then "LEN byte of T1 protocol is tool big to fit in IOBuf."

Give the person some leeway here, folks. They are not native English speakers, okay? So I presume he meant "too big."

And then he says, "Don't listen anymore IO line" -- the IO line is the input/output line. This is the data line we are listening to -- "but immediately log the info instead. This step is optional."

I'll explain what's happening there in a second.

What he does is he says, "Clear LEN and APDU_Index

to ensure no buffer hanky-panky (more robust!)"

Well, the interesting thing about this is relooking at the code, it's actually only really necessary to clear one of these, but this guy cleared two, so I think that's where the comment "more robust" comes from, okay? So he wasn't just closing the door. He was throwing the deadbolt and padlocking it as well.

BY MR. STONE:

Q Can you explain --

(Interruption in the proceedings.)

BY MR. STONE:

Q Can you explain exactly why that is the case?

A I'm sorry, I don't understand the question.

Q Well, are there two things going on or one thing?

A I see. Okay. I understand the question.

So obviously we have something first out here, but when you look at the code, you can see that he's clearing not just the LEN variable, but also the APDU Index variable. And from what I can see in the code, it was only necessary to clear one of them. That was all you needed to do, but this guy said, "I'm really going to check," all right? "There is no way we are getting past this," okay? "So I'm going to do two things to shut it down."

Now, here is the next thing, "Set one more bit in E011." Well, after you've worked with the system for a

while, you'll recognize E011 as being part of what's called the OTP area. OTP stands for one-time program vault. So the interesting thing about this portion of the memory is you can change it from 1 to 0, but you can never change it back, okay?

And so what this code does, here, is it takes the variable location E011 and forces one of the bits in it from a 1 to a 0. That is called "marking the card." In other words, because this is not a reversible process, you now put an indelible mark on that card that "Hey, somebody tried to overflow the buffer. The chances are it was a bad guy."

So you've done two things, now. You've literally stopped them overflowing the buffer, but you've also detected and recorded the fact that the attempt was made, and hence, that they are probably a bad guy.

Q And what did you do with that information?

A The importance of that information, now, is you can generate what's called an electronic countermeasure. So electronic countermeasure is something that comes down through the satellite, and it's a little program that essentially says, "Hey, is location E011 marked, yes or no? If it is, do something horrible to the card; disable it, destroy it," whatever, okay?

And so what NagraVision deployed in February 2000 was really a one, two punch. It was a patch, close the buffer

overflow, mark the card, and then an ECM, which comes through and says, "We are going to destroy anybody that attempts to use this buffer overflow."

Q Now, we've seen the effectiveness of the patch in the animation. Did you see other evidence that the patch was affected?

A Yes, I did.

Q And what was that?

A The first thing that I saw were the so-called Conus e-mails.

Q And do you have a graphic of those?

A Right.

So earlier on in my testimony, I mentioned that I read five ring binders, each that was this thick (indicating), okay? These are the Conus e-mails. And what these are is Mr. Conus sending reports back to NagraVision on the status of piracy.

February 22nd, 2001, this is what he reported, "Secured, VIP rights in same cards, blocked cards not affected by update." So this was just a few days after the patch had gone down, okay?

In December 11th, 2001, he reported, "Hole closed. Some cards have blocker software."

We go to August 26th 2002, same comment.

January 29th, 2003, same comment.

June 16, 2003, same basic comment.

Q Now, there is a reference on here to this blocker software. Did you see evidence that the blocker code did not have a large impact?

A Yes, I did.

Q And do we have a slide?

A Before I show you the slide, ladies and gentlemen, let me explain what a blocker code is, okay? A blocker code is a piece of code that is designed to prevent a card receiving that patch, okay? So if the card can't get patched, it's still vulnerable. So if you're a member, because NagraVision waited until the Nipper postings to start developing a patch, they gave the pirates a seven-week window. And the pirates knew that this buffer overflow vulnerability was what they call "unstable." In other words, it was easily fixed. And so what the pirates started to do is work like mad on trying to develop a blocker code that would prevent cards from taking the patch that they knew was coming.

Well, this is what Mr. Conus had to say on March 2nd, 2001, concerning the blocker codes: "The hacking is stalling, it seems. The number of hackers who still have non-updated cards is very limited, and therefore, the blocker code that has been published cannot be widely used."

Now, my interpretation of that is Mr. Conus is saying,

"Yeah, there is a few pirates still running our cards behind blocker code, but don't worry about them, they are very limited."

Q Now, we talked earlier about the countermeasure that accompanied the patch.

A Yes.

Q And did you see evidence regarding the effectiveness of that countermeasure?

A Yes, I did.

Q And do you have a slide of that?

A Mr. Guggenheim, who was CEO of NagraStar, was asked about the effectiveness of this one, two punch, the patch, and the follow-up electronic countermeasure. And this is what he had to say:

"And so anyone who is trying to overwrite the stack would now have a marked card, correct?"

Answer: "Correct."

"Permanently marked?"

Answer: "Correct."

"And then you'd come in with the kill shot later and zap all those cards that had been marked, correct?"

"Yes."

There's then a few more questions where he's asked about it, and he agrees at the end:

"Was it categorically effective?"

Answer: "Yeah."

Q Now, we've heard some testimony, though, that there were other flavors that supposedly were not affected by the patch or countermeasure; are you aware of that testimony?

A Yes, I have.

Q Do you agree with that testimony?

A No, I do not.

Q Why not?

A Well, to explain this, ladies and gentlemen, I'd like to give you a football analogy, okay? Let's say my hometown team, the Baltimore Ravens are in town to play the San Diego Chargers, okay? Ravens are a really good team. They've got lots of plays, lots of flavors of things they can do. They can razzle-dazzle you 17 ways.

If you are the San Diego Chargers, you look at that and go, "We've got a tough game on our hands here. How can we guarantee that Baltimore doesn't do all these different things to us?"

"Well, if you lock them in the dressing room, I think you can guarantee that Baltimore isn't going to beat you," okay?

Now, obviously, in the world of the NFL, locking your opponents in the -- in the dressing room is frowned upon. But in the world of satellite TV piracy, okay, keeping the guy off the field, or in this case, stopping him getting

into the card, prevents him from running all his different flavors of tricks. If he can't get in, he can't do anything.

Q Now, there was some discussion about glitching?

A Yes.

Q Was glitching mentioned at all in the Headend Report?

A No.

Q And is glitching a problem if you use the security sensors appropriately?

A Well, if they've chosen a card that had a functioning set of security sensors, it would not have been a problem, but in this case, yes, it was a huge problem.

Q Now, based on your examination of the patch code, the NagraVision reports, the documents you reviewed, the testimony that you reviewed, is it your opinion the patch was completely effective in preventing buffer overflow attacks?

A Yes.

Q Now, as you understand in this case, plaintiffs claim that there was a card swap that was necessary because of the December 2000 internet posting?

A Correct.

Q And do you recall when EchoStar made the decision to begin a card swap?

A I believe I've seen testimony from Mr. Ergen that said

the decision was made in 2002.

Q And from all the evidence you've examined, what was the status of the ROM 3 card in 2002?

A Hole closed, or card secured.

Q Have you seen any evidence that the card swap was necessary as a result of the December 2000 postings?

A Absolutely not.

Q Do you have an opinion whether a card swap was necessary as a result of the December 2000 internet posting?

A Yes, I do.

Q And what's your opinion?

A Well, the December 2000 postings exposed one vulnerability, of many, in this card, okay, the buffer overflow. That was patched. I've discussed how effective that patch was. The plaintiffs' own reports say that the hole is closed, the card is secured. So how is the card swap necessary based upon the NiPpEr2000 postings? It makes no sense.

MR. STONE: Thank you.

No further questions.

THE COURT: Do you want to take a recess before you start cross-examination?

MR. HAGAN: Please, your Honor.

THE COURT: All right. Why don't we do that.

You are admonished not to discuss this matter

amongst yourselves, nor form or express any opinion concerning the case.

We will start cross-examination in about 20 minutes.

(Recess.)

-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 30, 2008

_____

JANE C.S. RULE, U.S. COURT REPORTER

CSR NO. 9316