EXHIBIT C

IN THE UNITED STATES DISTRICT COURT

FOR THE EASTERN DISTRICT OF TEXAS

TYLER DIVISION

* * * * * * * * * * * * * * * * * *

BEDROCK COMPUTER TECHNOLOGIES, LLC, Civil Action No.

Plaintiff 6:09-CV-269-LED

vs.

SOFTLAYER TECHNOLOGIES, INC., ET AL.,

Defendants

* * * * * * * * * * * * * * * * * *

VIDEOTAPED DEPOSITION OF: ALEXEY KUZNETSOV

Taken before Isabelle Klebanow, Registered Professional Reporter and Notary Public, pursuant to Notice, at the offices of Goltsblat BLP Law Firm, Capital City Complex, Moscow City Business Center, 8, Prennenskaya Nab. Building 1, Moscow, Russia, on Thursday, January 27, 2011, commencing at 1:15 p.m.

A. Yes. Exactly.
Q. Okay. And did you testify that the lines below this comment remove records?
A. Yes.
Q. And did you also testify that, going back toward the top of the page, the rt_garbage_collect function also removed records?
A. Yes.
Q. Okay. So is it your understanding that there are two portions of code in rt_cache-add that remove records?
A. Yes.
Q. What was the purpose -- strike that.
Why did you put two different places -- strike that. I'm sorry. Let me try to ask the question more correctly.
Why did you include two sections of code in this function that each remove records?
A. Even though they do the same thing, they work quite differently.
Rt -- the first way, which does globals, can't remove entry, is much more efficient in the sense that it counts all the tables. So you can't write to the situation when one part of table is more empty and another has a lot of entries. So when we add an entry

to some part of hash table, this counts all the table to balance through all of it.
From the other hand, it is a slow process. We have to count through all the table. It's expensive. It's much cheaper to remove some entries which are already under hand. We add them to some hash line.
We already can count forward through the list to find some entries to remove right there, in hope then, if we are lucky, hash table size will not grow above the limit and the first function, rt_garbage_collect, will probably even never will be triggered.
Q. You referred to the rt_garbage_collect function as expensive, is that correct?
A. Yes.
Q. What do you mean by expensive?
A. It takes quite a lot CPU resources, CPU time, to count through all the hash table, which could be quite large in size.
Q. Okay. And just to back up just one more time. I think you testified that rt_cache-add calls rt_garbage_collect?
A. Yes.
Q. What does it mean to call rt_garbage_collect?
A. To use this function as part of work, and then it calls up -- it uses some piece of code from the

function. Then it goes to the original code.
Q. Okay. Right below the comment that says Cleanup duplicate and aged off entries, I think -- one, two -- three lines below it, there's a line that starts with the word While.
Do you see that?
A. Yes.
Q. Do you know what the purpose of that line is?
A. Yes. This line start of cycle which counts through hash table chain.
Q. And do you see, immediately below the while statement, do you see an opening brace?
A. Yes.
Q. What does that signify?
A. That everything which after this line until the closing brace is used -- is executed for each entry in this hash table chain.
Q. And where is the closing brace that you referred to?
A. Closing brace is on the next page, on the first line on the next page.
Q. Okay. And so, just to be clear, is the closing brace you're referring to on page 9314, and it's immediately above the restore_flags line?
A. Yes.

Q. So is it your testimony that, between the opening brace under the while loop and that closing brace at the top of 9314, that the code in between those two braces is executed in the while loop?
A. Yes.
Q. Okay. Going back to the rt_garbage_collect function, did you testify that that function is a garbage-collection process?
A. It's just a term. I -- I -- the term wasn't written by me. But probably it's just a term used by software programmers for -- garbage collection is a process of removal all unneeded data, and they link the sources to the system.
So, yes, I testified that the function rt_garbage_collect do garbage collection.
Q. And so is -- do you know if the code within the while loop is also performing garbage collection?
A. It is.
Q. So you have two garbage-collection routines in this function?
A. Yes.
(Exhibit No. 3 marked for identification.)
Q. Okay. I am handing you what has been marked as Exhibit 3 -- not just yet.

I am handing you what has been marked as Exhibit 3 (indicating). If you go by the Bates numbers at the bottom right side of the page, it begins with DEF00001013 and it ends on DEF00001043.
A. Yes.
Q. Do you recognize this document?
A. Yes. This is a doc content of the file. It is that file. It is a file of rt.
Q. Okay. And who -- who wrote this file?
A. It was written by many people. But, actually, I think 90 percent or more of this file was written by me.
Q. And when did you write -- strike that.
So do you see your name on the first page of this document?
A. Yes.
Q. So were you one of the people who wrote this document?
A. Yes.
Q. Okay. And when did you -- and so you wrote a portion of this document, is that right?
A. Yes.
Q. And other people wrote other portions?
A. Yes.
Q. When did you write your portion of this document?
A. I wrote it in 1995.

Q. Okay. And is the portion that you wrote the portion that we discussed earlier in connection with Exhibit 2, which is the patch?
A. Yes.
Q. Okay. So take a look at this, and I want to ask you if this is a true and correct copy of route.c from Linux version 1.3.52.
A. Yes. I think so. I cannot say it's exact. You just gave me this document.
Q. Okay. Mr. Kuznetsov, if you take a look at Exhibit 1, which is your declaration --
A. Yes.
Q. -- if you look at the second page of that, do you see paragraph 6 on the second page?
A. Yes.
Q. And it says that, Linux kernel version 1.3.52?
A. Yes. When I signed -- before I signed the document, I verified accurately that documents which were identified by those labels are genuine ones.
Q. Okay.
A. But I just -- you didn't tell me that it is that document.
Q. Okay. Do you see the numbers at the bottom right side of the first page?
A. Yes. I understand.

Q. Okay. And so those numbers on the bottom of the page correspond to the numbers in paragraph 6?
A. Yes.
Q. Okay. So -- and when you signed the declaration on December 15, 2010, you personally verified that this was -- that the version of Linux 1.3.52 --
A. Yes.
Q. -- represented by these pages is a true and correct copy?
A. Yes.
Q. Okay. Let's turn to page -- let's see -- to page 1036. And just let me know when you find that page.
A. Okay. I found it.
Q. Okay. Thank you. Do you see a line around the middle of the page that starts with, Static void rt_cache-add?
A. Yes.
Q. Okay. And is that the same function rt_cache-add that we discussed earlier today?
A. Yes.
Q. Okay. And do you know if any other portions of route.c use the rt_cache-add function?
A. Yes.
Q. Do you know what portions those are?

A. I can find it. A function -- for example, it's a function next to rt_cache-add function, ip_rt_slow_route. The function is responsible for creation of a new routing cache entry.
And then close to end -- it's page 1039 -- it calls rt_cache-add.
Q. Okay. So it's your testimony that ip_rt_slow_route calls the rt_cache-add function?
A. Yes.
Q. And, also, could you please turn to the page ending in 1032.
A. Okay.
Q. Do you see a function that says rt_redirect_1 at the top of the page?
A. Yes.
Q. And do you see, around the middle of the page, there's a statement that says rt_cache-add?
A. Yes, yes, yes.
Q. So what is happening at that line where it says rt_cache_add?
A. Rt_cache-add adds new cache entries, so it's called when we have to add new cache entry. Function rt_redirect_1 creates stockhold redirected entry.
It's special kind of entry which triggered by a network event that, when we send packet to some router,

Is that the same while statement that we discussed earlier in connection with your patch?
A. Yes.
Q. Okay. And so is this a while loop?
A. Yes.
Q. And can you identify the beginning and ending of that while loop?
A. Yes.
Q. Okay. Please do.
A. Beginning is a line which starts from the word While. It's a line which starts from word While. It immediately follows comment which we discussed which starts cleanup duplicated and aged off entries.
And this loop ends right before line containing restore flags.
Q. Just so the record is clear, the entire while loop is contained on page 1037, correct?
A. Yes.
Q. And so does this while loop continue walking through the records in the linked list that was accessed earlier?
A. Yes.
Q. And so does the while loop start over at the beginning of the linked list?
A. Not -- it continues scan the linked list.

Current head of linked list is already new. So we have -- this While continues scan through linked list, but new entries.
We don't want to delete these entries, so we continue the scan of table of linked list.
Q. Okay. Can you describe what happens within this while loop?
A. Yes. We check -- we scan through all the chain, and analyze each routing cache entry which we encounter for the full condition.
First, we check that this entry is too old and not a reference right now. In this case, we can delete this entry. This entry is expired. And also we check that this entry points to the same destination which -- to the same destination as entry which we have just added. So we must delete this entry.
So if one of these conditions is satisfied, we unlink current entry from hash chain, decrease size of cache size to follow the fact that we deleted some entry, and destroy this routing entry.
The function rtfree destroys current entry, and after this we continue the scanning table for link.
Q. Okay.
THE VIDEOGRAPHER: This is the end of tape No. 2. Off the record. Time is 3:22.

(Short recess taken.)
THE VIDEOGRAPHER: On the record. This is cassette No. 3, and the time is 4:35.
Sorry. 3:35.
Q. Okay. Mr. Kuznetsov, thank you. We are back on the record.
And shortly before we took our break, you -- we were discussing the while loop contained on page 1037, in rt_cache-add.
If you look at the line below the cli in the while loop, the if statement, could you -- what does that if statement do?
A. This statement checks the current routing cache entry can be deleted. It's either expired or duplicate.
Q. Okay. And if the record is expired, what happens next?
A. If the record is expired, it is deleted from hash chain. Routing cache size is decreased. Then it's destroyed, and we proceed with current table of the hash chain.
Q. Very good. And does any of the lines of code in the while loop go back to the head of the linked list?
A. No.
(Exhibit No. 4 marked for identification.)

Q. I would like to introduce another exhibit. This is Exhibit 4.
Mr. Kuznetsov, I'm handing you what's been marked as Exhibit 4 (indicating). It is -- if you go by the number at the bottom of the page, it starts with DEF. It starts with DEF00009321 --
A. Yes.
Q. -- and continues through DEF00009349.
A. Yes.
Q. Do you recognize this document?
A. Yes. It is the same file. Probably it is of -- according to the number before --
(Witness peruses documents.)
A. I don't see reference to this number.
Q. Let's also use -- let's use the bottom number then that says KS-DEF. And it starts with --
A. Page 6 -- yes -- I don't see it. Oh, yes. I found it. It's statement No. 3 in my declaration. Yes, it is the same file.
Q. And what is this document?
A. It is the same file.
Q. So is it route.c?
A. Yes.
Q. And which version of Linux?
A. It's from kernel 1.3.42.

Q. Okay. And is this a true and correct copy of the route.c from above?
A. Yes. I identified this.
Q. Okay. And does this -- does Exhibit 4, this version of route.c, include the code you wrote that was included in the patch that was Exhibit 2 today?
A. Yes.
Q. And if you turn to the page ending in 708 -- using the bottom set of numbers, it starts with KS-DEF and ends with 708, do you see that page?
A. Yes.
Q. And do you see, near the top of that page, rt_cache-add?
A. Yes.
Q. And is this the same rt_cache-add function that we saw in your patch that was Exhibit 2 in today's deposition?
A. Yes. Exactly the same.
Q. Okay. Do you know if this version of route.c was made publicly available at any time?
A. Yes. It was publicly available as Linux version 1.3.42.
Q. Okay. Do you know when this version of route.c was made publicly available?
A. Actually, I cannot tell this exactly. But I

verified before that it was issued on November 17 of 1994. November 16.
Q. So what year -- excuse me. What date was it available on?
A. November 16, 1994.
Q. 1994?
A. 5. I'm sorry. 1995.
Q. So just to make the record clear, this version of route.c was publicly available on November 16, 1995?
A. Yes.
Q. And how -- and how could someone have accessed this document in 1995?
A. He could go to FTP site where Linux kernels are stored to download huge file which contains all the files related to Linux kernels.
This file was numbered 1.3.42, and you could unpack the file and to fetch the file route.c from there.
Q. Okay. Do you know anyone who accessed this file in 1995?
A. The question is more complicated. Yes, of course. It was -- at least it was me, Alan Cox, Linus Torvalds. So --
But I can't -- many people who downloaded that file, but I cannot tell who exactly it was.

Q. But --
A. I know it was thousands of people.
Q. Okay. And you downloaded that file in 1995?
A. Yes. Particularly I downloaded exactly at the day when it was published.
Q. Okay. Very good. Let's introduce another exhibit.
(Exhibit No. 5 marked for identification.)
Q. I am handing you what has been marked as Exhibit 5. It is -- let's see.
If we use the numbers at the bottom right corner of the page, it starts with DEF00007865 and ends at DEF00007899.
And do you recognize this document?
A. Yes.
Q. And what is it?
A. It's almost the same document. It's different previous one in few lines. It's document from a little later kernel version, as I testified before, kernel version 1.3.51.
Q. So is this a true and correct copy?
A. Yes, it is true, true accurate copy of file route.c from Linux version 1.3.51.
Q. Okay. And turning to the page ending in 7892,

you see rt_cache-add at the top?
A. Yes.
Q. Okay. And is this the same rt_cache-add function that is in Exhibit 2?
A. Yes.
Q. Okay. And was this document ever publicly available?
A. Yes. It was publicly available as part of kernel 1.3.51.
Q. And when was it first publicly available?
A. On or around December -- December 27, 1995.
Q. And how could someone have accessed this document in 1995?
A. Could unload this document as part of Linux source archive version by No. 1.3.52, after unloading the FTP site.
Q. And do you know anyone who accessed this document in 1995?
A. Yes. It was me; Linus Torvalds, who created the document, at least Alan Cox, to whom this patch was submitted. But I downloaded it.
Q. Okay.
(Exhibit No. 6 marked for identification.)
Q. Handing you now what has been marked as

Exhibit 6. It's -- using the numbers at the bottom center of the page, it starts with DEF00008567 and continues through DEF 00007601.

Do you recognize this document?

A. Yes.

Q. And what is this document?

A. It is the same document, slightly modified. Some parts were fixed. Little features were added. And this document was published as part of Linux kernel version 2.0.1.

Q. Okay. And was this document publicly available?

A. Yes.

Q. And when was it first publicly --

A. This document was publicly available around July 3, 1996.

Q. Okay. And how would someone have accessed this document in 1996?

A. Anyone could download Linux source archive version 2.0.1, to unpack it, and to fetch file route.c from there.

Q. Do you know if anyone accessed this document in July 1996?

A. I downloaded the document.

I don't know personally anyone who did the same, but I can surely say that this document, all the archive

was created and put to FTP site by Linus Torvalds. So he obviously accessed the document.

Q. Okay. Let's turn to the page ending in 8592.

A. Okay.

Q. And do you see line 1299? It says rt_cache_add?

A. Yes.

Q. What is on line 1299?

A. It's beginning of the first rt_cache-add.

Q. We've discussed rt_cache-add earlier today.

Is this the same function -- are there any differences between the rt_cache-add in this document and the rt_cache-add we discussed in Exhibit 2?

A. No. It is exactly the same function. The file was changed a little, but no changes were made for this function.

Q. Okay.

(Exhibit No. 7 marked for identification.)

Q. I am now handing you what has been marked as Exhibit 7. Using the numbers at the bottom center of the page, it starts with DEF00008602, and it continues through DEF8605.

Do you see that?

A. Yes.

Q. Have you seen this document before?

A. Yes.

Q. And what is it?

A. It's a so-called header file for routing cache. It contains some definition comment shared by file route.c and another component of Linux kernel.

Q. Okay. And so this is a true and correct copy of the route.h file from Linux version 2.0.1?

A. Yes.

Q. And was this document publicly available?

A. Yes. This document was made publicly available simultaneous with file route.c.

Q. Okay. And do you know if anyone accessed this document in 19- -- July 1996?

A. Yes. I did.

Q. Okay. If you turn to the second page, at line 65, and you see the word struct rtable?

A. Yes.

Q. What does that mean?

A. It's -- this structure contains all the data for a particular routing cache entry. It contains all the data, including destination where to send this packet and linkage in hash table.

Q. Okay. And you see line 67? It says struct rtable *rt_next.

A. Yes.

Q. What does that line do?

A. This is pointer to the next element in hash table, which is in the linkage in routing hash table which we discussed before.

Q. Right. So is it a pointer to the next element in the linked list?

A. Yes.

Q. Okay. And if you look then on line 98, it says, extern struct rtable. That's where the line starts. And then it says, ip_rt_hash_table.

What does this line mean?

A. This line declares -- describes hash table which contains pointers to hash chains consisting of struct rtable, which we just discussed.

Q. Okay. And so were records corresponding to routing cache entries stored in that hash table?

A. Yes.

Q. Okay. All right. Mr. Kuznetsov, are any defendants in this case paying you for your time in connection with this deposition?

A. No.

Q. Do you have as -- do you have any opinion on the validity of US patent No. 5,893,120?

A. Yes.

MR. CASSADY: Object to form.

Q. Okay. So the affidavit was written and controlled by Mr. Absher, right?
THE INTERPRETER: (Russian).
A. Yes.
Q. And after reviewing it, you signed it, right?
A. Yes.
Q. But you didn't make any edits to the draft that Mr. Absher sent you, isn't that true?
A. If something was wrong there -- actually, the document contains only facts. If some facts were not true, I would edit it.
But, actually, I did not have to do any changes. All the facts are correct, to all I know.
Q. So you signed the original version of the draft that Mr. Absher sent you?
THE INTERPRETER: (Russian).
A. Yes.
Q. Okay.
A. What is original? I got a PDF file. I printed it. I signed it. I sent it to Alton Absher.
Q. Okay. Now, Mr. Kuznetsov, we know you talked to some of the defendants in this case. We know you talked to some of the attorneys for the defendants in this case.
But have you ever attempted to contact Dr. Nemes,

who's the inventor on the patent in this case?
MR. ABSHER: Object to form.
A. No.
Q. So you've never spoken with Dr. Nemes the inventor on the patent in this case?
MR. ABSHER: Object to form.
A. No. I know definitely, no.
Q. So have you ever spoken with any attorneys from Bedrock about this case?
A. No.
Q. So is it fair to say that you've only heard the story from the side of the defendants in this case?
MR. ABSHER: Object to form.
A. Yes. First of all, yes, I heard the story only from the -- only from defendants.
But the patent is filed in US patent system. I read it independently, not under guidance of anybody.
Q. Okay. So isn't it true, sir, that, as of December of 2010, you had reviewed the patent in this case at least twice?
A. Yes.
Q. And now, isn't it true, sir, that the code written by you does not actually collide with the afore-mentioned patent because your code uses quite different techniques, isn't that true?

MR. ABSHER: Object to form.
A. The code how it existed since 19- -- since 1997 and until 2008, when it was patched by Eric Dumazet, indeed does not collide with the patent. That is my statement. It doesn't collide with the patent.
But the piece of code which was added by Eric Dumazet indeed collide with the patent. But this code is just an insignificant improvement of my code which I wrote in 1985.
And then, due to its utter inefficiency, it was removed from the kernel at some point. For at least five years, it wasn't used in Linux kernel. Not five. More.
But the idea remained the same, and we knew how to -- at some point, we knew how to improve it. So, actually, it is seen from the code. The code which was added by Eric Dumazet is exactly the same place which was used in my code dated about 1995, and does almost exactly the same thing.
Q. Okay. Mr. Kuznetsov, let me ask you a very particular question.
So the code written by you in 1995 does not actually collide --
A. No. My code --
Q. Sir, sir, Mr. Kuznetsov, let me finish my

question. One moment. I apologize. Let me finish my question, okay?
It is true, sir, that the code written by you, in 1995, does not actually collide with the patent in this case, because your code uses quite different techniques? Isn't that true, sir?
MR. ABSHER: Object to form.
A. No. It is not true. My code, written in 1995, directly collide with the patent. Actually, it used exactly the same technique which was described in the patent up to letter.
But this code was inefficient, and I removed that code. This code was not in Linux kernel for many years. And, at this time, Linux kernel didn't collide with the patent because it used quite different expiration technique.
Then, my idea from 1995, which existed in Linux kernel at least for two years, was returned in another incarnation and applied to Linux kernel by Eric Dumazet.
At this point it started to collide again.
Q. Now --
THE VIDEOGRAPHER: Sorry. Just a minute, Jason. I have four minutes of tape. Shall I change it right now?
MR. CASSADY: Yes.