
بسم الله الرحمن الرحيم

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

پروژه کارشناسی مهندسی کامپیوتر

عنوان پروژه:

تحلیل پروتکل ارتباطی پایگاه داده اوراکل (TNS) و توسعه یک پروکسی ساده برای آن
Analyzing Oracle database communication protocol (TNS) and developing a simple proxy

دانشجو:

محمدحسین هوشمند

۹۸۳۶۱۳۰۶٤

بهار و تابستان ۱٤۰۲

فهرست مطالب

چکیده

سامانه‌های پایگاه داده مثل SQL Server یا Oracle مثل اغلب نرم‌افزارهای سرور، یک یا چند پروتکل ارتباطی مخصوص به خود دارند که نرم‌افزارهای سرویس‌گیرنده از طریق این پروتکل‌ها به آنها متصل شده و عملیات مورد نظر خود مانند پرس‌وجو را انجام می‌دهند. البته به واسطه‌ی وجود درایورهایی مانند ODBC[1] یا ADO[2] و موارد مشابه، برنامه‌نویسان با جزییات این پروتکل‌ها درگیر نمی‌شوند و از طریق این درایورها با پایگاه ارتباط می‌گیرند. اما این درایورها درنهایت از پروتکل مخصوص همان پایگاه داده استفاده می کنند. برای مثال پروتکل استفاده شده در SQLServer ، TDS[3] نام دارد و پروتکل مورد استفاده در پایگاه داده اوراکل ،TNS[4] است.

پروتکل شبکه ی TNS یک پروتکل اختصاصی برای اتصال نظیر به نظیر دستگاه‌های سرویس‌گیرنده به پایگاه داده‌ی اوراکل است. این پروتکل در داخل لایه NET8 هسته‌ی اصلی ارتباط با پایگاه داده‌ی اوراکل را شکل می‌دهد.

برای پروتکل SQLServer مستندات کافی توسط مایکروسافت منتشر شده و در دسترس می‌باشد اما در مورد پروتکل TNS مستندات رسمی منتشر نشده است. اطلاعاتی که از این پروتکل در اختیار است اغلب به صورت مهندسی معکوس جمع‌آوری می‌شود.

هدف این پایان‌نامه تحلیل پروتکل TNS و تهیه مجموعه مستنداتی از خصوصیات آن است. در کنار این موضوع، هدف ثانویه پروژه، پیاده‌سازی یک پروکسی ساده برای این پروتکل می‌باشد. این پروکسی بین سرویس‌گیرنده و سرور پایگاه داده اوراکل قرار می‌گیرد و می‌تواند محتوای بسته‌های منتقل شده را بازبینی کند.

این پروکسی قادر است اطلاعات مهم نشست را در قالب لاگ ثبت کند و برای آزمایش ارتباط مشتری و میزبان مورد استفاده قرار می‌گیرد.

[1] Open Database C
[2] ActiveX Data Objects
[3] Tabular Data stream
[4] Transparent Network Substrate

پروکسی مربوطه در محصول رایمون شرکت پیام‌پرداز استفاده می‌شود که یک محصول از خانواده‌ی PAM[1] می‌باشد که وظیفه‌ی اصلی چنین نرم‌افزارهایی، کنترل دسترسی‌های ادمین شبکه و پایگاه داده می‌باشد.

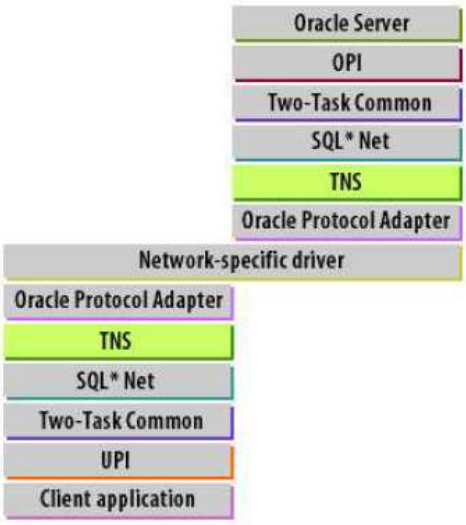
[1] Privilege Access Management

فصل اول

پروتکل TNS

۱-۱- معرفی

پروتکل TNS یک پروتکل اختصاصی از سوی شرکت اوراکل می‌باشد که وظیفه‌ی آن، مدیریت و انجام اقدامات لایه‌های میانی اتصال پایگاه داده_مشتری (طبق مدل OSI) است. این پروتکل درصدد برقراری اتصالی قابل اعتماد فارغ از محدودیت‌هایی شامل تعداد مشتریان، تعداد پایگاه‌های داده، نوع ساختار و آرایش شبکه و مشخصات سخت‌افزاری و نرم افزاری ماشین‌ها می‌باشد. متاسفانه از جزییات این پروتکل برخلاف پروتکل مشابه خود در پایگاه داده‌ی SQL (پروتکل TDS) اطلاعات دقیق و رسمی از سوی شرکت اوراکل منتشر نشده است و این شرکت اطلاعات را به صورت کلی و کاربرد محور به مهندسان عرضه کرده است.



شکل (۱-۱): جایگاه پروتکل TNS

۱-۲- معماری اوراکل و جایگاه پروتکل

از پروتکل TNS به عنوان قلب تپنده‌ی معماری اوراکل نام برده می‌شود. جایگاه ویژه‌ی این پروتکل در بالای لایه‌ی فیزیکی، موجب شده است وظایف حیاتی و مهمی بر عهده‌ی این لایه قرار بگیرد. همانطور که پیشتر اشاره شد، معماری اوراکل بر اساس ساختار شناخته‌شده‌ی OSI طراحی و پیاده‌سازی شده است. این ساختار که بسیار شناخته‌شده است، ماشین‌ها را به تعدادی لایه‌ی طبقه‌بندی شده از نرم‌افزاری تا سخت‌افزاری برنامه‌نویسی می‌کند. ساختار OSI این امکان را فراهم می‌کند که از طریق پروتکل‌های رایجی چون TCP-IP در لایه‌های نزدیک به لایه‌ی فیزیکی، ارتباط قابل اعتمادی میان مشتریان و پایگاه‌های داده برقرار شود. برای شناخت بهتر پروتکل TNS لازم است نگاه دقیق‌تری به ساختار اوراکل و لایه‌های آن داشت تا به اهمیت جایگاه این پروتکل پی برد[۱][۲].

۱-۲-۱- لایه‌ی نمایش

این لایه نزدیک‌ترین و قابل لمس‌ترین لایه به کاربر می‌باشد و معادل لایه‌ی Application در معماری OSI است. با این تفاوت که در سمت ماشین مشتری(سرویس‌گیرنده) از آن به عنوان لایه‌ی مشتری و در سمت ماشین سرویس‌دهنده به عنوان سرور یاد می‌شود. این لایه تحت عنوان لایه‌ی نرم‌افزاری مدل اوراکل نامبرده می‌شود و حاوی برنامه‌ها و درایورهایی است که درخواست و پاسخ ماشین‌ها را دریافت می‌کند. در این لایه مشتری می‌تواند درخواست خود را به صورت پرسش SQLPlus در کنسول وارد کند.

در سمت ماشین مشتری، پرسش به لایه ی پایین‌تر ماشین یعنی لایه‌ی OCI ارسال می‌شود و بالعکس در ماشین سرویس‌دهنده، درخواست از سوی لایه‌ی پایین‌تر ماشین میزبان یعنی OPI دریافت می‌شود. این لایه معمولا به خاطر شباهت وظایفی که با لایه‌ی زیرین خود دارد قابل ادغام می‌باشد

۱-۲-۲- لایه‌ی مشتری- سرور

این لایه در ماشین مشتری به نام OCI و در ماشین سرویس‌دهنده به عنوان OPI نام‌گذاری می‌شود. در سمت مشتری، درخواست ها به صورت پرسش SQLPlus به لایه‌ی OCI وارد می‌شود و پس از پردازش درخواست، خروجی به صورت یک مجموعه توابع استاندارد اوراکل تبدیل می‌شود. بالعکس در سوی پایگاه داده، این لایه پاسخ OPI را دریافت و به صورت یک داده‌ی یا تراکنش قابل درک به لایه‌ی سرور می‌رود.

۱-۲-۳- لایه‌ی TTC

معمولا عواملی چون اختلاف سیستم‌عامل در دو ماشین باعث می‌شود که در مجموعه‌ی کاراکترهای مورد استفاده، اختلافاتی وجود داشته باشد که کاملا طبیعی است. همچنین ممکن است در نوع داده‌هایی که مورد پذیرش دو سیستم است، تناقض به وجود آید. لذا برای رفع این مسئله، لایه‌ی TTC[1] وظیفه اصلاح و برطرف کردن این دو مشکل را دارد. مخصوصا در زمانی که اتصال مشتری و سرور برقرار می‌شود، نقش این لایه در برقراری توافقات اولیه بر سر این دو نقطه‌ی بحرانی پررنگ‌تر می‌شود. پس از اتصال، این لایه به وظیفه‌ی تبدیل نوع و تغییر مجموعه‌ی کاراکترها در صورت نیاز می‌پردازد. بنابراین قابل درک است که برای مدیریت این دو وظیفه‌ی مهم، نام‌گذاری این لایه منحصربه‌فرد باشد.

از این لایه به عنوان آخرین لایه‌ی حوزه‌ی نرم‌افزار نام برده می‌شود و واسطی میان لایه‌های شبکه و لایه‌ی مشتری- سرور به حساب می‌آید.

۱-۲-٤- لایه‌ی NET8

لایه‌ی NET8 تمامی وظایف و اعمال مربوط به لایه ی شبکه را به عهده می‌گیرد. این لایه خود به سه زیرلایه‌ی (NET Interface , Routing/Naming/Auth , TNS) دیگر قابل تقسیم است. همانطور که قابل حدس است، این لایه وظایف مهم و حیاتی چون ایجاد و نگهداری اتصال، احراز هویت کاربران، مسیریابی بسته‌ها و سمدیریت

[1] - Two-Task Common

و ارسال خطاها را بر عهده دارد. در این لایه نقش پروتکل TNS پررنگ می‌شود. زیرا حجم زیادی از وظایف این لایه، توسط این پروتکل انجام و مدیریت می‌شود. این پروتکل در نقش یک میانجی بین پیغام‌هایی که از سمت لایه‌ی(OCI-OSI) و لایه‌ی Transport دست‌به‌دست می‌شود، ایفای نقش می‌کند. نکته‌ی حائز اهمیت این است که این ارتباط بدون درنظر گرفتن تفاوت‌های سیستم‌عامل، سخت‌افزار و نرم‌افزار انجام می‌شود.

به عنوان مثال، زمانی که اتصال بین مشتری و پایگاه داده برقرار می‌شود، پیش از تبادل درخواست، مشتری از طریق یک شنونده که نقش یک پروکسی را ایفا می‌کند(اطلاعات مربوط به سرورهای موجود در فایل شنونده ذخیره شده است)، آدرس سرور درخواست شده را برمی‌دارد یا اطلاعات مربوط به خود را برای سرور ارسال می‌کند. همچنین از طریق این پروتکل دو ماشین به توافق می‌رسند که طبق چه ورژنی از اوراکل ارتباط خود را آغاز کنند یا با چه پروتکل شبکه‌ای (مانند TCP/IP) اطلاعات را تبادل کنند. از دیگر وظایف لایه‌ی NET8 که مربوط به زیرلایه‌ی دوم آن است، مدیریت نام‌گذاری‌ها و فراهم ساختن مقدماتی برای تامین امنیت داده‌ها یا احراز هویت کاربران است.

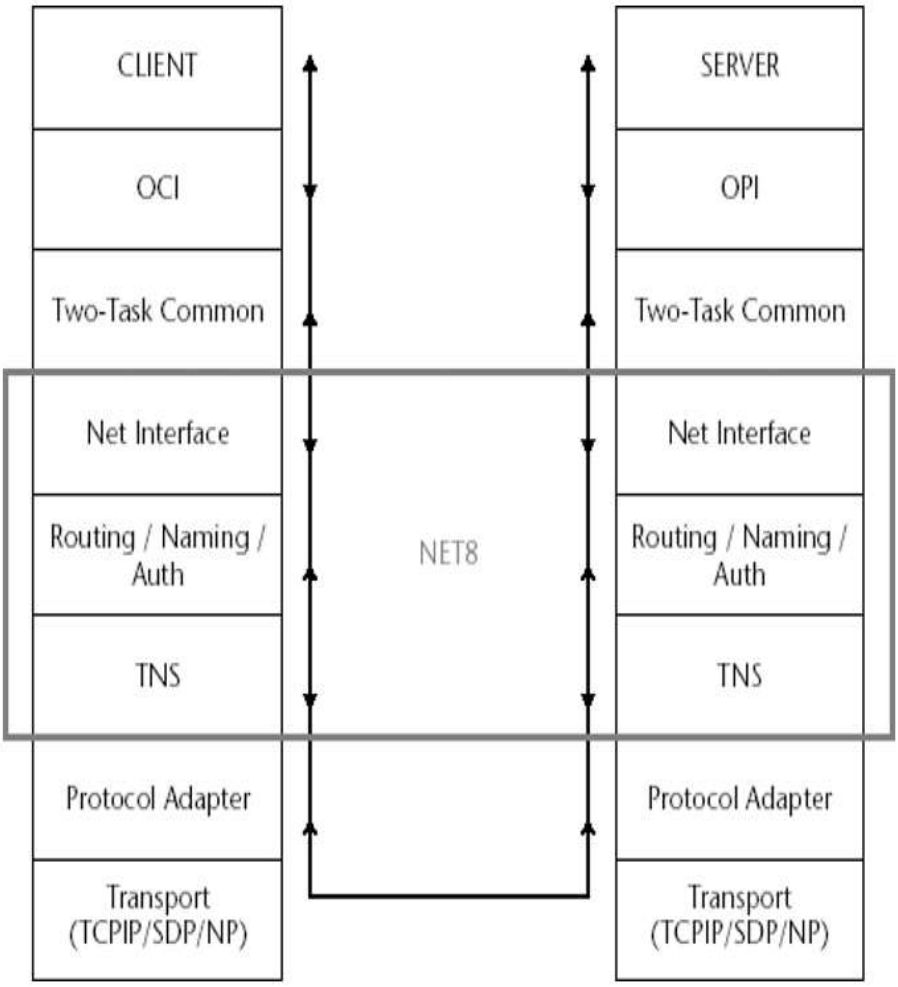
۱-۲-۵- لایه‌ی Protocol Adaptor

از این لایه به عنوان یک پنجره میان لایه‌ی انتقال و لایه‌ی NET8 یاد می‌شود. وظیفه‌ی مدیریت و تشخیص پیغام‌های خطای پروتکل‌ها از مهم‌ترین وظایف این لایه می‌باشد. همچنین این لایه شرایط لازم را برای استفاده از پروتکل با توجه به زیرساخت‌های سیستم‌عامل فراهم می‌کند.

۱-۲-٦- لایه‌ی انتقال

این لایه آخرین لایه‌ی معماری اوراکل می‌باشد و کار انتقال داده‌ها در این لایه صورت می‌گیرد. در این لایه عملکردها و عملیاتی که توسط پروتکل TNS تعریف شده است، در قالب یکی از پروتکل‌های رایج و استاندارد مانند TCP/IP، SDP، Named pipe یا TLS صورت می‌گیرد.

بدیهی است ماشین‌هایی که از زیرساخت‌ها و محصولات اوراکل استفاده می‌کنند، باید شرایط و امکانات لازم برای برقراری اتصال از طریق پروتکل‌های ذکر شده را داشته باشند.



شکل (۱-۲): معماری اوراکل

۱-۳- نحوه عملکرد

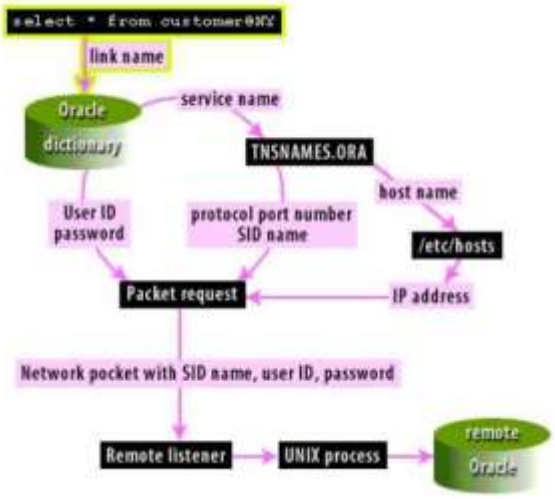
این پروتکل زمینه‌ی اتصال ایمن و قابل اعتمادی را بین دو ماشین مشتری و سرویس‌دهنده برقرار می‌سازد. در ادامه نحوه عملکرد پروتکل در فرآیند پردازش یک درخواست را بررسی خواهیم کرد. مراحل به این شرح است[٤]:

۱-۳-۱- احراز هویت کاربر:

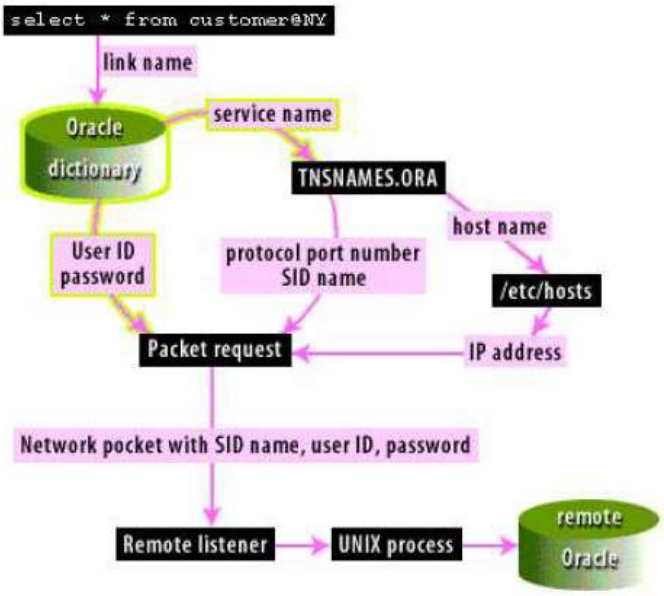
ابتدا کاربر مورد نظر باید نام‌کاربری و رمزعبور خود را وارد نماید. این اطلاعات به واحدی به نام oracle dictionary ارسال می‌شود و مورد بررسی قرار می‌گیرد. درصورتی که کاربر از قبل هویت خود را مشخص کرده باشد، از این مرحله صرف نظر می‌شود.

۱-۳-۲- بررسی درخواست سرویس:

درخواست مشتری که به صورت یک پرسش SQLplus نوشته شده است، به واحد Oracle dictionary ازسال می‌شود. Oracle dictionary بر اساس درخواست مشتری، تشخیص می‌دهد که او به چه سرویسی نیاز دارد. در ادامه Oracle dictionary نام سرویس مورد درخواست را به عنوان خروجی باز می‌گرداند. فرآیند بازیابی نام سرویس می‌تواند در حین احراز هویت کاربر انجام شود. در این صورت فرآیند بازیابی نام سرویس پیش از شروع پرسش و پاسخ انجام می‌شود.



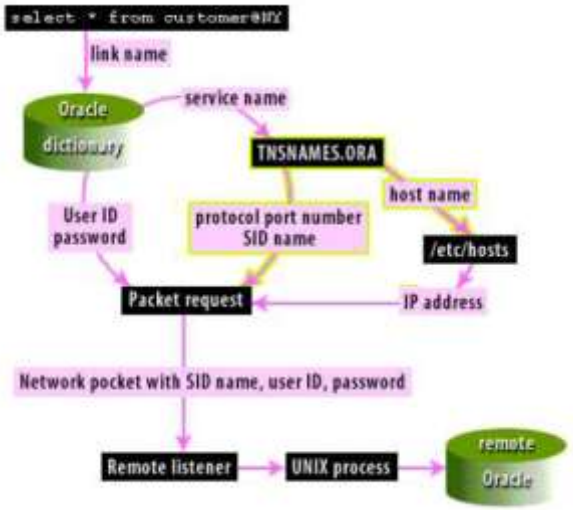
شکل (۱-۳): ارسال پرسش به Oracle dictionaary



شکل (۱-٤): خروجی‌های Oracle dictionary

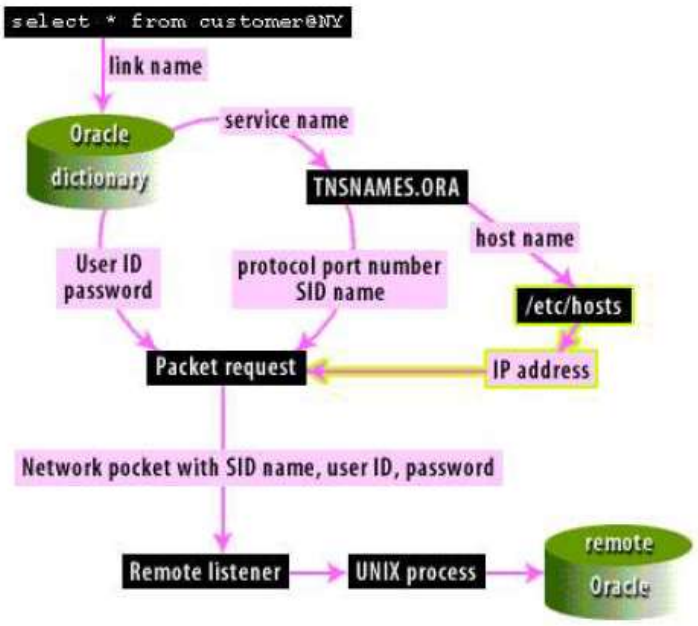
۱-۳-۳- بازیابی اطلاعات سرویس‌دهنده:

در ادامه نام سرویس استخراج شده، در فایل مخصوصی به نام tnsnames.ora در ماشین مشتری مورد بررسی قرار می‌گیرد. این فایل مشخصات هر نام سرویس به همراه آدرس و پورت و نام میزبان سرویس‌دهنده را در خود ذخیره کرده است. مشتری این فایل را بررسی می‌کند و اطلاعات ذکر شده را استخراج و بازیابی می‌کند.



شکل (۱–۵): استخراج آدرس و پورت

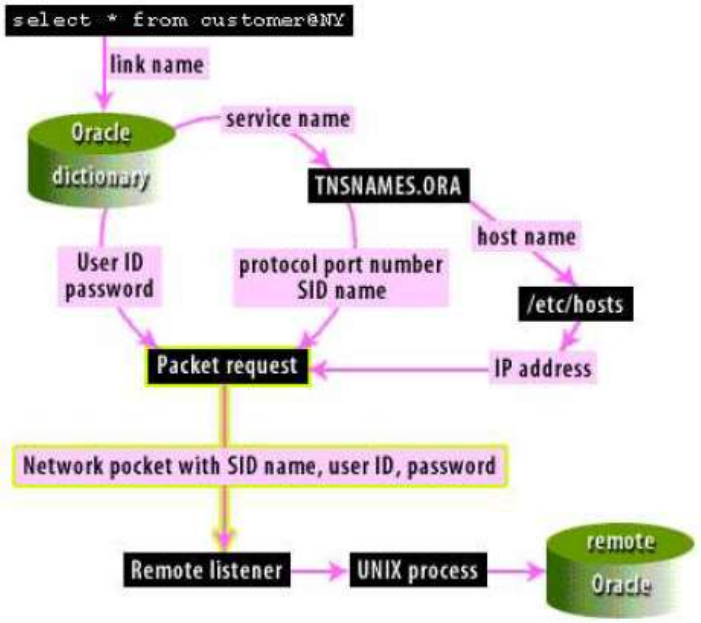
در برخی موارد مشتری می‌تواند با استفاده از نام سرویس، به پوشه‌ی /etc/hosts رجوع کند و آدرس IP را بازیابی کند.



شکل (۱–٦): افزودن آدرس به بسته

۱-۳-٤-ایجاد بسته‌ی پرسش:

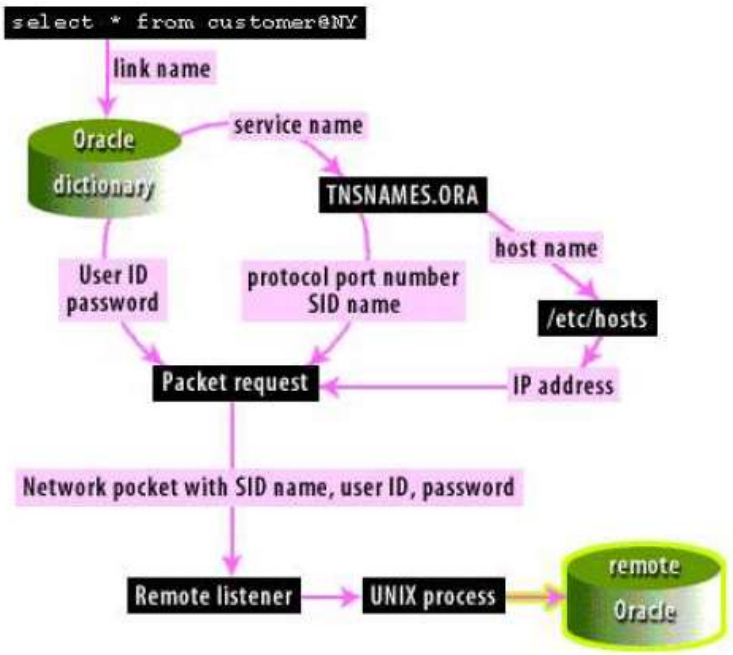
در این مرحله با استفاده از داده‌های استخراج شده مانند آدرس IP، شماره‌ی پورت، شناسه‌ی سرویس‌دهنده، نام کاربری و رمزعبور، بسته‌ی پرسش ایجاد می‌شود. در ادامه این بسته از طریق یکی از پروتکل‌های ارتباطی شبکه مانند TCP/IP به ماشین سرویس دهنده ارسال می‌شود تا ادامه‌ی مراحل در ماشین سرویس‌دهنده انجام شود.



شکل (۱-۷): افزودن آدرس به بسته‌ی پرسش

۱-۳-۵- دریافت بسته پرسش:

در ماشین سرویس‌دهنده، ابزاری به نام شنونده قرار دارد که به درخواست‌های پورت(پیش‌فرض ۱۵۲۱) گوش می‌کند. شنونده بررسی می‌کند که این درخواست را باید به کدام پایگاه داده ارسال نماید. این ابزار پس از اعمال سلسله‌ای از عملیات یونیکس، به پایگاه داده‌ی اوراکل متصل می‌شود و بسته را به دست آن می‌رساند. پس از این‌که شنونده اتصال بین مشتری و سرویس‌دهنده را برقرار کند، سرویس‌دهنده می‌تواند پاسخ را به مشتری ارسال کند.در ادامه شنونده کنار می‌رود و این ارتباط تا زمان پایان نشست ادامه پیدا می‌کند.



شکل (۱-۸): ارسال پرسش به سرویس اوراکل

۱-۳-٦- توقف و پایان نشست:

پس از اتمام فرآیند پرسش و پاسخ، زمانی که مشتری قصد دارد ارسال پرسش را متوقف کند، بسته‌ای را برای توقف فرایند ارسال می‌کند و به این ترتیب به این چرخه پایان می‌دهد.

۱-٤- مزایای پروتکل

پروتکل TNS ویژگی‌های مثبت قابل توجهی دارد که در این قسمت به چند مورد مهم آن اشاره می‌شود[۳]:

۱- سهولت درک شبکه برای ماشین‌ها و برنامه‌ها: شبکه‌هایی که در سیستم اوراکل استفاده می‌شود، از آنجایی که از مدل OSI توسعه پیدا کرده‌اند، برای ماشین‌ها شفاف و قابل درک هستند. پروتکل TNS هم از این قاعده مستثنی نیست و به گونه‌ای در ساختار شبکه گنجانده شده است که برخی از وظایف شبکه را با در نظر

گرفتن ساختار OSI یه صورت فشرده انجام می‌دهد. پس در نتیجه برنامه‌ها می‌توانند بدون نگرانی از ساختار و ویژگی‌های شبکه با پایگاه‌های داده‌ی اوراکل کار کنند.

۲- مدیریت اتصال و ارتباطات: پروتکل TNS با درنظر گرفتن تعدد ماشین‌های مشتری و سرویس‌دهنده، از طریق روش Connection pooling پیشرفته سعی بر مدیریت اتصالات دارد. در این صورت تعداد زیادی مشتری می‌توانند با تعداد محدودی پایگاه داده اتصال برقرای کنند و به این صورت سرعت و کارایی ارائه‌ی خدمات افزایش می‌یابد.

۳- به‌کارگیری ابزار شنونده: با توجه به تعدد در ارائه‌ی پایگاه‌های داده توسط یک سرویس‌دهنده، وجود ابزاری برای شناسایی پایگاه داده‌ی درخواست شده برای سرویس‌دهنده احساس می‌شود. لذا در این فرآیند، نقش شنونده پررنگ می‌شود. شنونده در ماشین سرویس‌دهنده قرار گرفته است و در انتظار پرسش و پیغام‌های مشتریان می‌نشیند. شنونده درخواست را براساس اطلاعاتی مانند نام سرویس دریافت می‌کند و پایگاه داده‌ی مناسب را به مشتری متصل می‌کند. در واقع شنونده مسئولیت بررسی و شناسایی اتصال‌های گوناگون به سرویس‌دهنده را برعهده می‌گیرد.

از دیگر وظایف مهم شنونده، پایان دادن به اتصالات بلااستفاده است. اتصالاتی که در آن هیچگونه تبادل پیام خاصی صورت نمی‌گیرد. این کار منجر به کاهش فشار بار به شبکه می‌شود.

٤- سهولت آدرس‌دهی: پروتکل TNS برای دسترسی به یک پایگاه داده‌ی خاص، نیازی به دانستن آدرس دقیق و اطلاعات خاص پایگاه داده ندارد. صرفا آگاهی از نام سرویس و مجموعه‌ای از "توصیفات اتصال" برای انجام این کار کافی هستند.

۵- سازگاری و انعطاف‌پذیری: این ویژگی به خدمات مبتنی بر سرویس‌های اوراکل این امکان را می‌دهد که مشتریان و سرویس‌دهندگان، نگرانی از بابت نوع ساختار شبکه و تفاوت سیستم‌عامل ماشین‌ها، سخت‌افزار یا نرم‌افزارها نداشته باشند. در نتیجه امکان گسترش خدمات و سرویس‌دهی افزایش پیدا می‌کند.

٦- امنیت: پروتکل TNS امکان ارائه‌ی روش‌های مختلف رمزنگاری و امنیتی را برای سرویس‌های مبتنی بر اوراکل فراهم گرده است. به عنوان مثال از پروتکل SSL/TLS برای ایجاد اتصال امن استفاده می‌شود و از الگوریتم تغییریافته‌ی DES برای رمزنگاری رمزعبور استفاده می‌کند[۵].

۱-٤- محدودیت‌ها و چالش‌های پروتکل

پروتکل TNS در کنار نقاط عطف خود، با چالش‌ها و محدودیت‌هایی همراه است که در ادامه چند مورد مهم آن برسی می‌شود[۳]:

۱-اختصاصی بودن: همانطور که پیشتر گفته شد، این پروتکل مخصوص سرویس‌ها و سیستم‌های اوراکل است و یک پروتکل عمومی نمی‌باشد. بنابراین امکان برقراری این پروتکل بر روی سایر سیستم‌های پایگاه داده وجود ندارد.

۲-دشواری و پیچیدگی ساختار: شرکت اوراکل به دلیل اختصاصی بودن این پروتکل و ضرورت محرمانه بودن جزییات وظایف آن، اطلاعات دقیقی از عملکرد و شگردهای امنیتی آن ارائه نداده است. این موارد به همراه دشوار بودن پیاده‌سازی شبکه‌ی مبتنی بر اوراکل نسبت به سایر محصولات مشابه از دیگر نکات قابل اشاره است. بنابراین برای مدیریت و نگهداری از شبکه‌های مبتنی بر اوراکل، نیاز به دانش و آشنایی کافی با جزییات قسمت-های مختلف آن وجود دارد.

۳-احتمال کاهش کارایی و عضو متورم: همانطور که در معماری اوراکل اشاره شد، چند لایه از مدل OSI در قالب یک لایه‌ی مهم به نام NET8 خلاصه شده است و از قضا بخش اعظمی از قراردادها و اصول این لایه، توسط پروتکل TNS انجام می‌شود. وظایفی که شامل رمزنگاری، برقراری اتصالات، مدیریت خطاها و... می‌باشد. به عنوان مثال شنونده که بخشی از وظایف پروتکل را انجام می‌دهد، یک ابزار متورم محسوب می‌شود و در صورت هجوم می‌تواند موجب اختلال مجموعه شود.

٤-ضرورت تنظیم صحیح مولفه‌های امنیتی: با وجود اینکه اوراکل از لحاظ امنیتی، از مکانیزم‌ها و ترفند-های مناسب و سختگیرانه استفاده می‌کند، لازم است به درستی تنظیمات امنیتی بر روی شبکه اعمال شود و نسنجیده این تنظیمات تغییر نکند. در غیر این صورت خطر حمله افزایش پیدا می‌کند

۵- تاثیرپذیری در مقیاس‌های بزرگ: در مقیاس‌های بزرگ، زمانی که تعداد فراوانی مشتری بخواهند از سرویس‌دهندگان خدمات بگیرند ممکن است مدیریت اتصال‌ها و رسیدگی به درخواست‌ها کند صورت بگیرد. در نتیجه استفاده از روش‌ها و راه‌حل‌های بهینه‌سازی در شبکه‌های با مقیاس بزرگ‌تر ضروری است.

۱-۵- ضعف‌های امنیتی

علیرغم شیوه‌ها و ترفندهای مهم امنیتی اوراکل، نیاز به اشاره به موارد مهمی است که اگر نادیده گرفته شوند، می‌تواند بسیار خطرآفرین باشد و به قیمت حمله به سیستم تمام شود[۳].

۱-تنظیم نادرست شنونده: شنونده یکی از مهم‌ترین نهادهای سرویس‌دهنده‌ی اوراکل است و سهم زیادی از وظایف پروتکل را بر دوش می‌کشد. همان‌طور که پیش‌تر اشاره شده، وظیفه‌ی مدیریت و برقراری اتصال مشتری به پایگاه داده‌ی خاص بر عهده‌ی این نهاد است. بدیهی است که این نهاد می‌تواند به یک نقطه‌ی آسیب‌پذیر و حساس برای سیستم به حساب آید. به عنوان مثال ممکن است شنونده به گونه‌ای تنظیم شود که به ماشینی اجازه‌ی برقراری اتصال به پایگاه داده‌ی خاصی را بدهد. حالت دیگر آن است که شنونده به گونه‌ای تنظیم شود که اتصالات و درخواست‌ها را به پایگاه داده‌ی حمله‌کننده ارجاع بدهد.

۲-ایرادات نرم‌افزاری: هر نرم‌افزاری ممکن است حفره‌هایی داشته باشد و این ایرادات هیچوقت به صفر نمی‌رسد. پروتکل TNS هم از این قاعده مستثنی نیست. لذا تضمینی وجود ندارد که این پروتکل بتواند امنیت کامل سیستم را تامین کند. با این حال چیزی که بر این نگرانی بیشتر دامن می‌زند، باز نبودن اطلاعات و جزییات این پروتکل است. به طوری که عیب‌یابی و یافتن راه‌حل برای مهندسانی که از این سیستم‌ها استفاده می‌کنند دشوار می‌شود.

۳- شبکه‌ی ناامن: چنانچه سیستم اوراکل در شبکه‌ای استفاده شود که از لحاظ امنیتی آسیب پذیر باشد، مهاجمان می‌توانند از طریق یک اتصال غیرمجاز به شنونده متصل شوند که این اتفاق می‌تواند شامل هرگونه خرابکاری شامل مسموم‌کردن اطلاعات و درخواست اتصال به پایگاه داده‌ای ویژه باشد.

فصل دوم

پیاده‌سازی سرویس‌دهنده و مشتری اوراکل

۲-۱- مقدمه

در فصل قبل به توضیح اطلاعاتی درباره‌ی پروتکل TNS پرداخته شد. همانطور که گفته شد اطلاعات موجود از سوی اوراکل تاحد زیادی محدود و محرمانه است و ابعاد زیادی از جزئیات پروتکل اوراکل هنوز ناشناخته باقی مانده است. با این حال مهندسان همواره در تلاشند تا با انجام آزمایش‌ها و بررسی سناریوهای گوناگون، موارد ناشناخته‌ی پروتکل را کشف و مهندسی‌معکوس کنند.

۲-۲- نصب ماشین مجازی

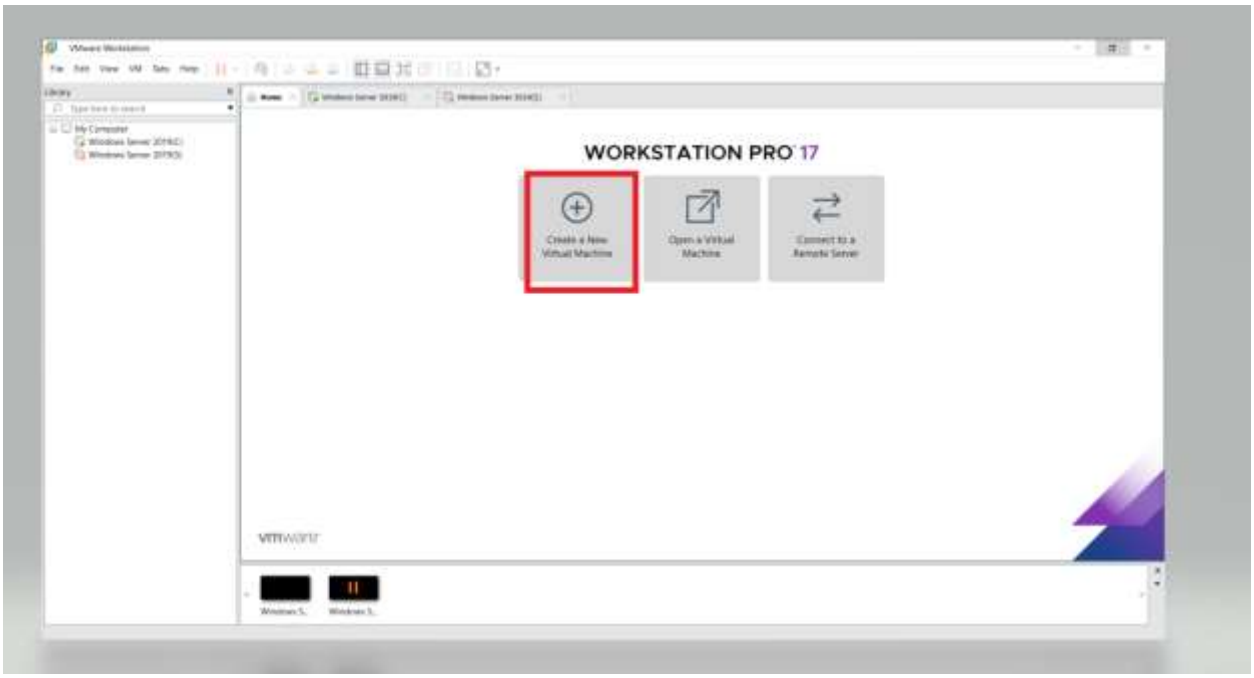
برای کشف و بررسی پروتکل TNS، نیازی به اتصال فیزیکی دو کامپیوتر وجود ندارد. به سادگی با کمک نصب دو ماشین مجازی، می‌توان به انجام آزمایش و تحلیل پروتکل پرداخت. در این راستا، یک ماشین مجازی نقش سرویس‌دهنده را ایفا می‌کند و ماشین مجازی دوم نقش مشتری یا سرویس گیرنده را ایفا می‌کند.

برای انجام آزمایشات، تفاوت چندانی میان دو سیستم عامل ویندوز سرور ۲۰۱۹ و لینوکس وجود ندارد و اوراکل با هر دوی این سیستم‌های عامل تعامل خوبی دارد. از آنجایی که در طول آزمایشات ممکن است مدام ماشین جدیدی راه‌اندازی و نصب شود، انتخاب ویندوزسرور ۲۰۱۹ به دلیل نصب آسان‌تر می‌تواند به سرعت

کار بیافزاید. با این حال نوع سیستم‌عامل طبق بررسی‌های انجام شده، در روند آزمایشات و پیاده‌سازی اوراکل تفاوت آشکاری ایجاد نمی‌کند.

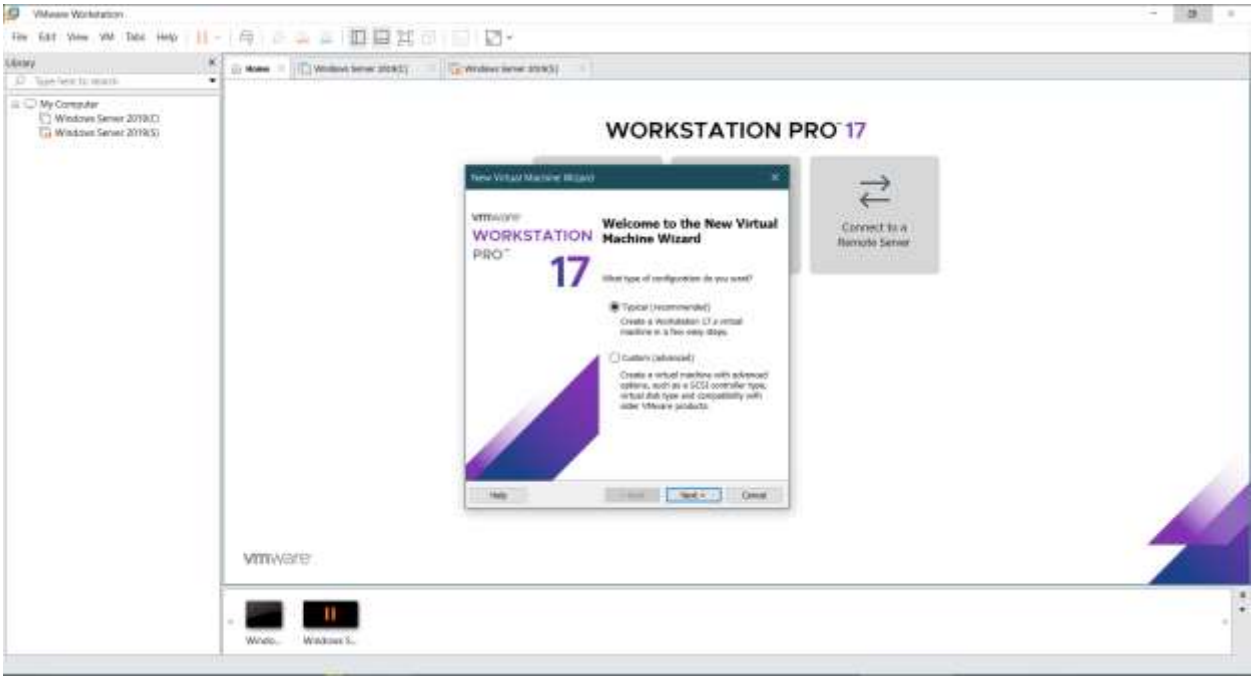
برای ایجاد یک ماشین مجازی، دو مجازی‌ساز Virtualbox و Vmvare بسیار پرکاربرد هستند. در ادامه به طور اجمالی نصب سیستم‌عامل ویندوز سرور ۲۰۱۹ برروی Vmvare اشاره می‌شود:

۱- در گام اول بر روی گزینه‌ی Create new virtual machine کلیک شود.



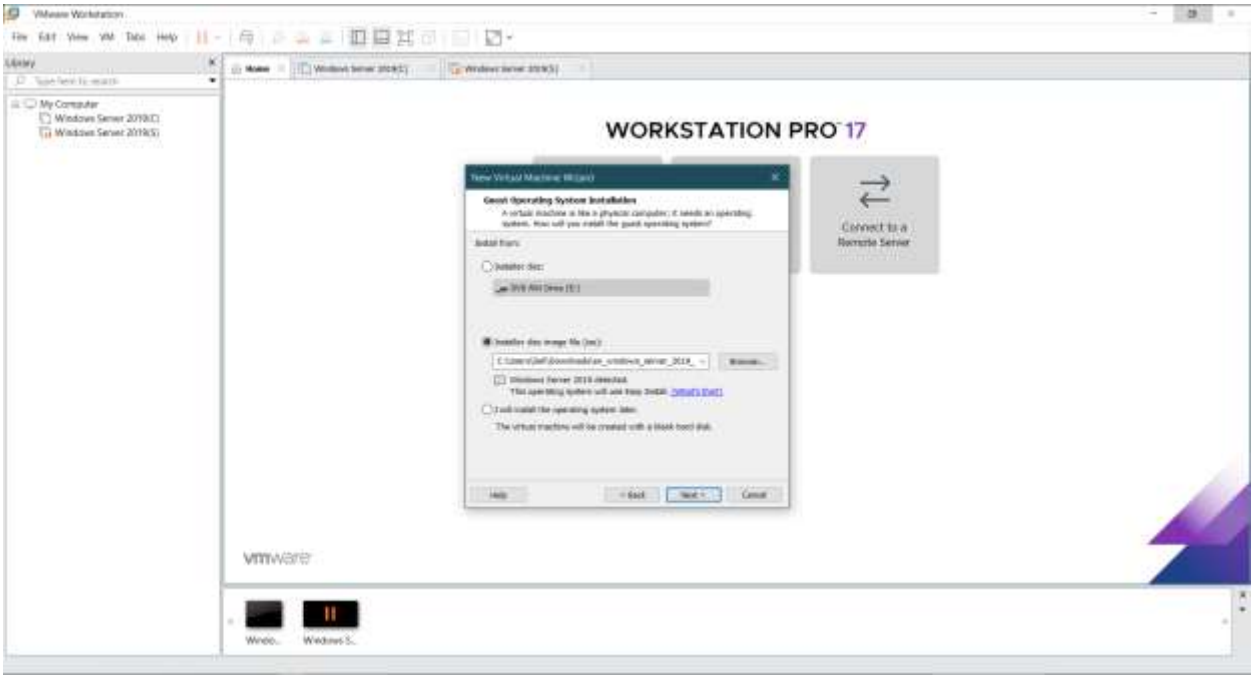
شکل (۲-۱)

۲- بر روی گزینه‌ی next کلیک شود.



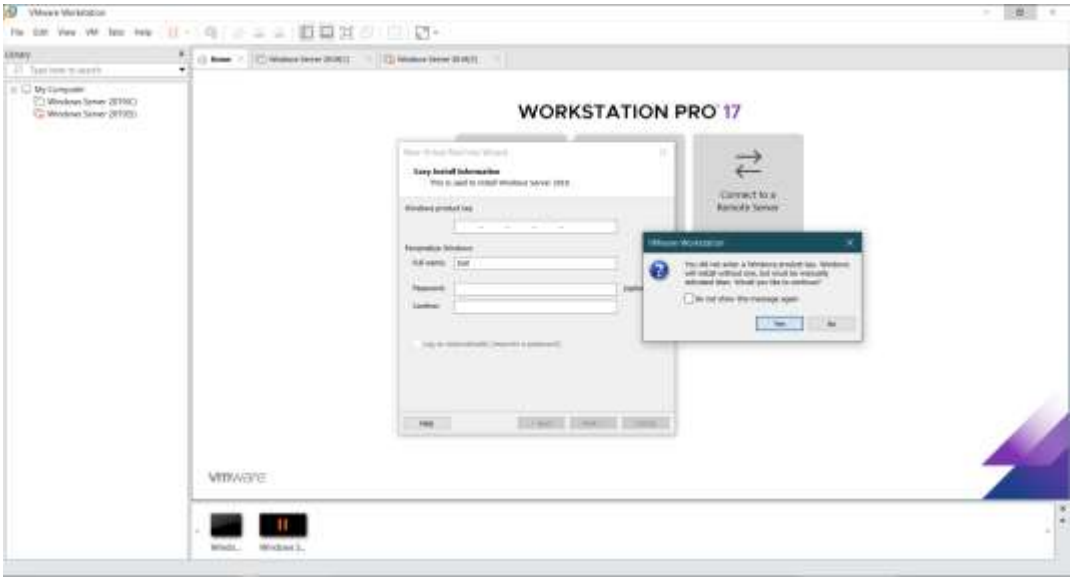
شکل (۲-۲)

۳- با کلیک بر روی گزینه‌ی browse فایل iso ویندوز سرور(که پیشتر باید دانلود شده باشد) انتخاب شود.



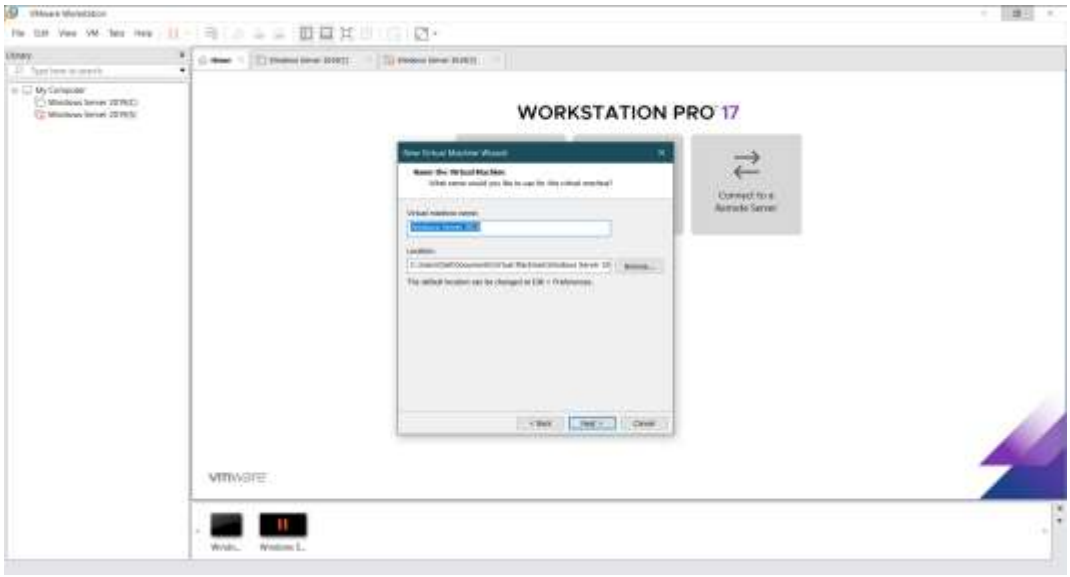
شکل (۲-۳)

٤- بر روی گزینه‌ی next کلیک شود. در صورت مواجه با پیغام عدم وارد کردن windows product key، پیغام نادیده گرفته شود.



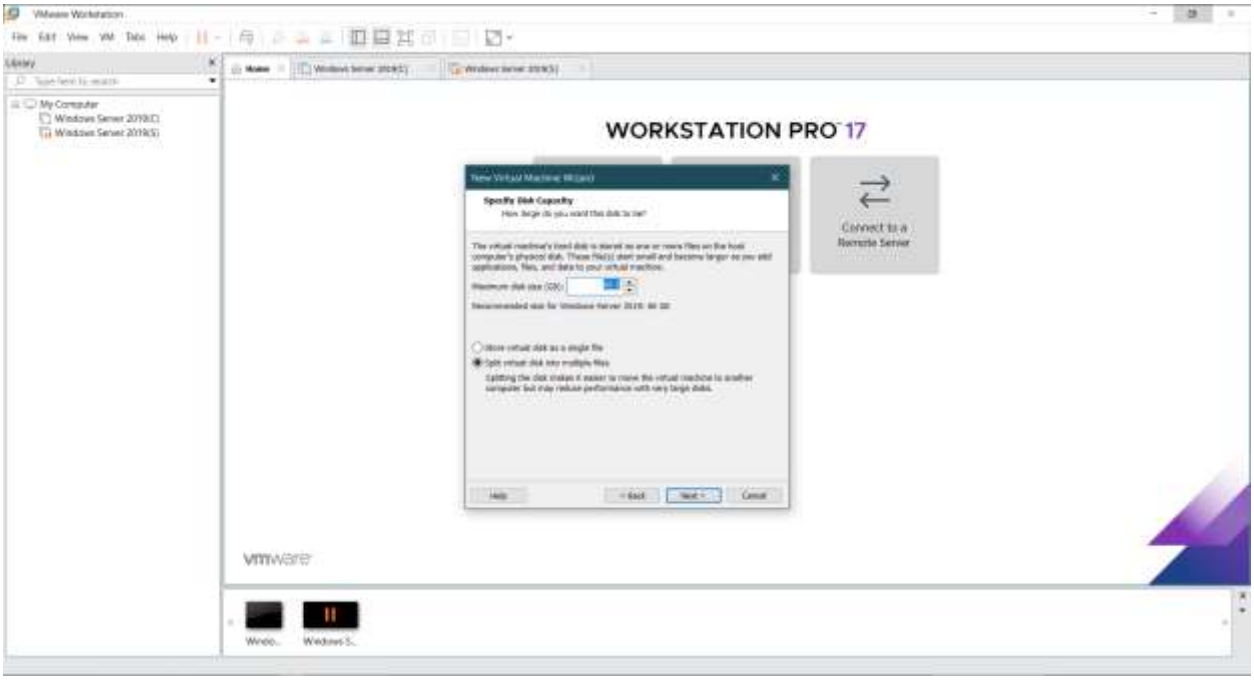
شکل (۲-٤)

۵- برای انجام آزمایشات نیاز به حداقل دو ماشین مجازی است. یکی به نام ویندوز سرور ۲۰۱۹ (S) در نقش سرویس‌دهنده و دیگری را به عنوان ویندوز سرور ۲۰۱۹ (C) در نقش مشتری نام‌گذاری می‌شود.



شکل (۲-۵)

٦- با توجه به حجم آزمایشات، تخصیص ۳۰ گیگ حافظه داخلی(چه برای سرویس‌دهنده و چه مشتری) کافی به نظر می‌رسد.



شکل (۲-٦): تعیین حافظه برای ماشین

۷- پس از آن بر روی گزینه‌ی بعدی و گزینه‌ی پایان کلیک شود. در ادامه روال عملیات نصب ویندوز سرور انجام می‌شود.

۲-۳- نصب نرم‌افزارهای مورد نیاز

پس از راه‌اندازی دو ماشین مجازی سرویس‌دهنده و مشتری، زمان نصب دو نرم‌افزار اوراکل‌سرور و اوراکل کلاینت فرا می‌رسد. اوراکل‌سرور سیستم سرویس‌دهنده‌ای است که می‌تواند چندین پایگاه داده را به طور همزمان مدیریت و سازمان‌دهی کند. البته این نرم‌افزار علاوه بر بر سرویس‌دهنده، چندین ابزار مدیریت و کارآمد دیگر را با خود نصب می‌کند. نرم‌افزار اوراکل‌کلاینت امکانات مدیریتی اوراکل‌سرور را ندارد و صرفا امکان ارسال درخواست را برای کاربر فراهم می‌نماید.

مهم‌ترین ابزاری که امکان ارسال درخواست را فراهم می‌کند، ابزار کنسول SQLPlus است که هم در اوراکل-سرور و هم در اوراکل‌کلاینت موجود می‌باشد. چنانچه امکان نصب اوراکل‌کلاینت فراهم نشود، می‌توان برای انجام تحقیقات، بر روی هر دو ماشین، اوراکل‌سرور را نصب کرد. در این صورت می‌توان در سمت ماشین مشتری درخواست‌ها را به ماشین سرویس‌دهنده ارسال نمود.

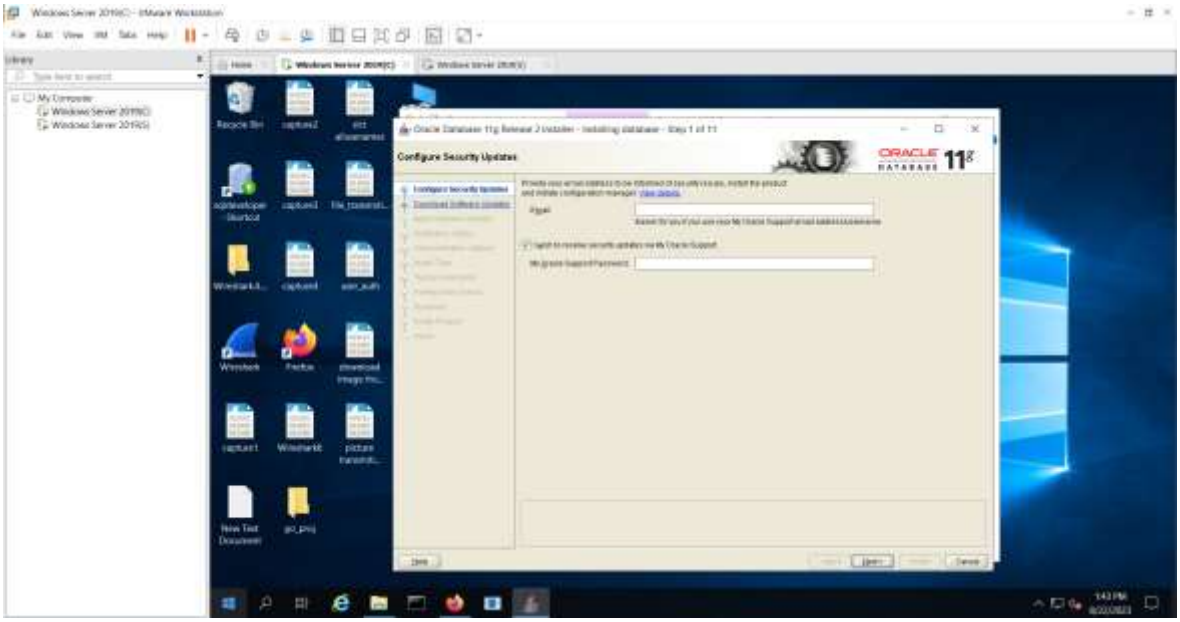
۲-٤- نصب اوراکل سرور

در اولین مرحله، لازم است اوراکل سرور از سایت اوراکل دانلود شود.

https://www.oracle.com/in/database/technologies/oracle-database-software-downloads.html

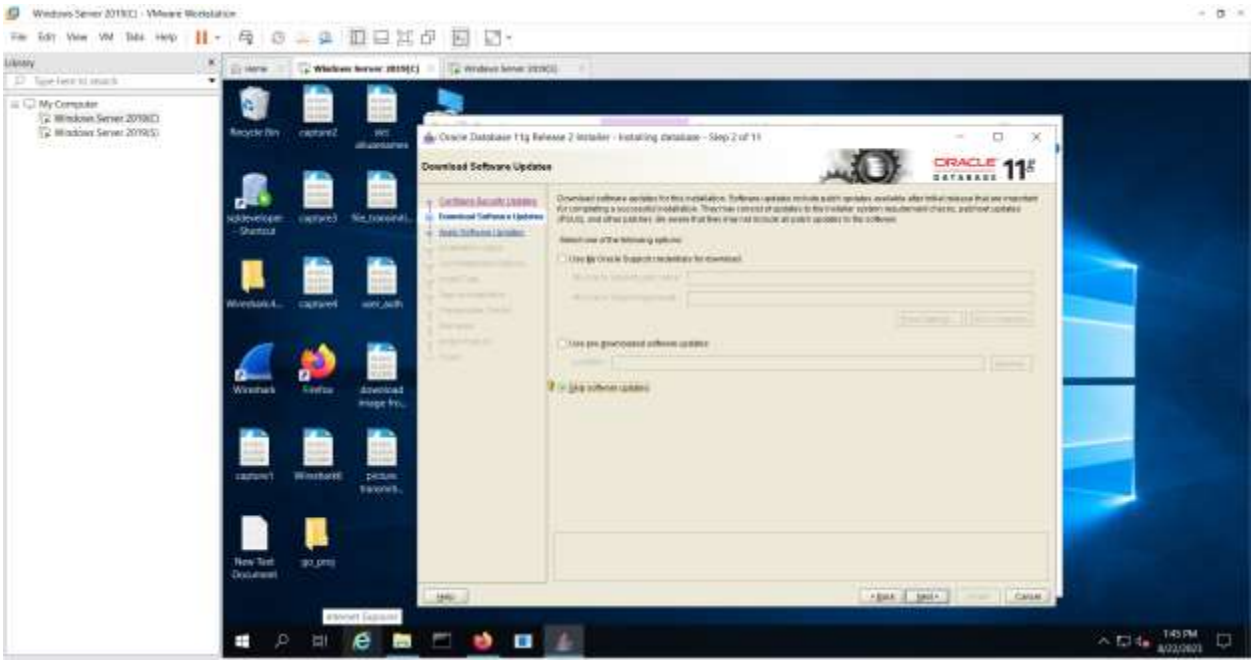
ممکن است دانلود این نرم افزار به علت مسائل ثبت‌نام یا تحریم دشوار باشد. بنابراین ترجیحا از نسخه‌ی لوح فشرده‌ی آن استفاده شود. نسخه ای که در این آزمایشات استفاده می‌شود، نسخه‌ی Oracle 11.2.0.2 Database x64 یا Oracle11g است.

۱- در ابتدا از صفحه زیر گذر کرده و در صورت بروز هشدار، نادیده گرفته می‌شود.



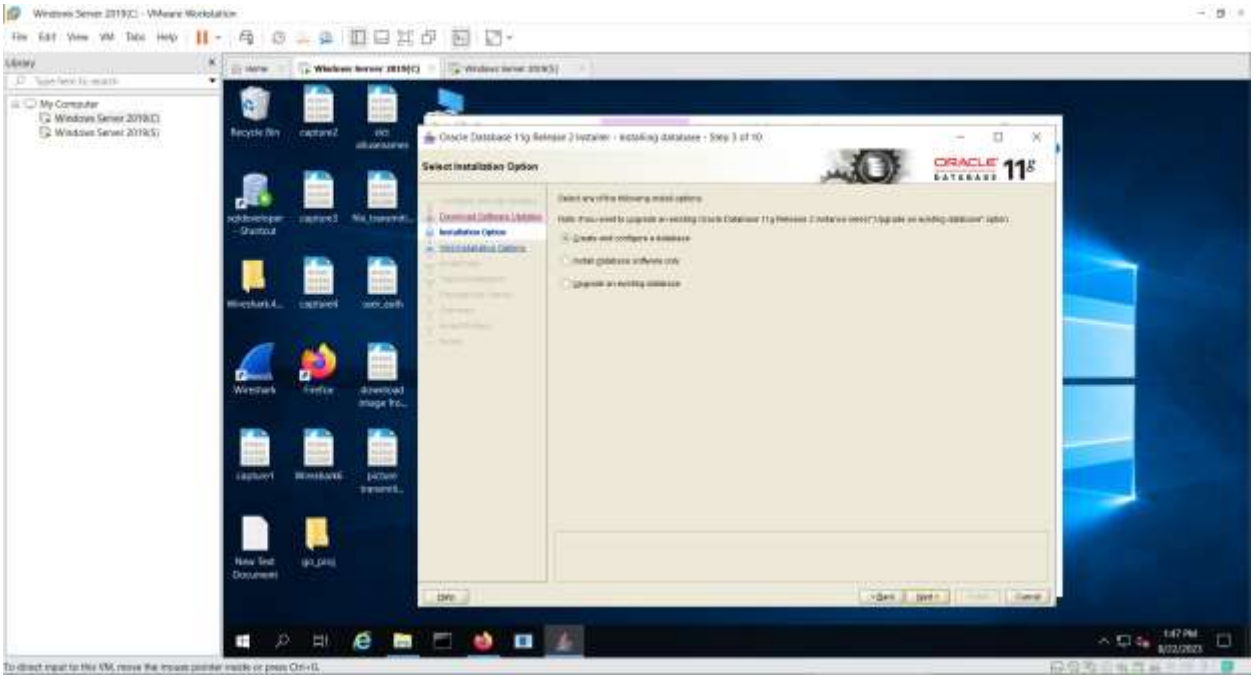
شکل (۲-۷)

۲- در صفحه بعد بر روی گزینه سوم کلیک کرده تا بروزرسانی‌های جدید نادیده گرفته شود.



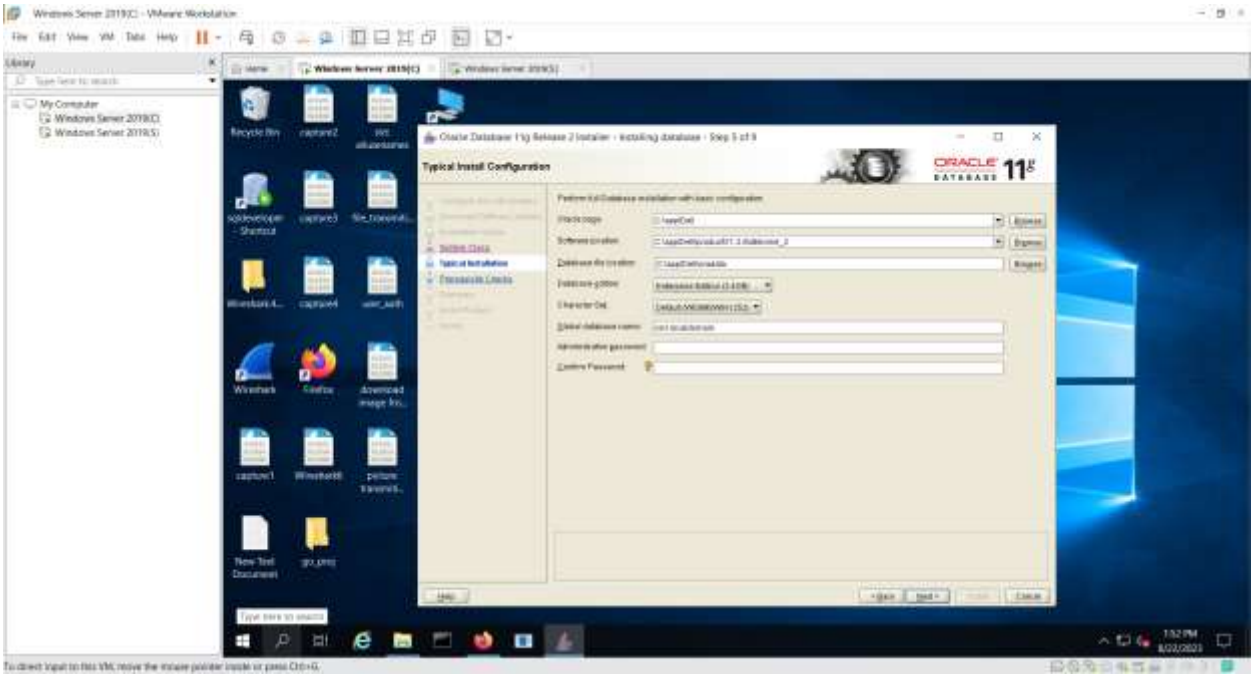
شکل (۲-۸)

۳- در این قسمت حالت نصب برنامه بر روی پیش فرض قرار داده می‌شود.



شکل (۲-۹)

٤- در ادامه صفحه‌ی بعد ظاهر می‌شود. ترجیحا بر روی مکان نصب فایل تغییری ایجاد نشود. در ادامه نامی برای دیتابیس جامع انتخاب شود. این نام می‌تواند به عنوان مثال orcls انتخاب شود. در قسمت بعد، به عنوان مدیر سرور، لازم است رمز عبوری انتخاب شود.



شکل (۲-۱۰)

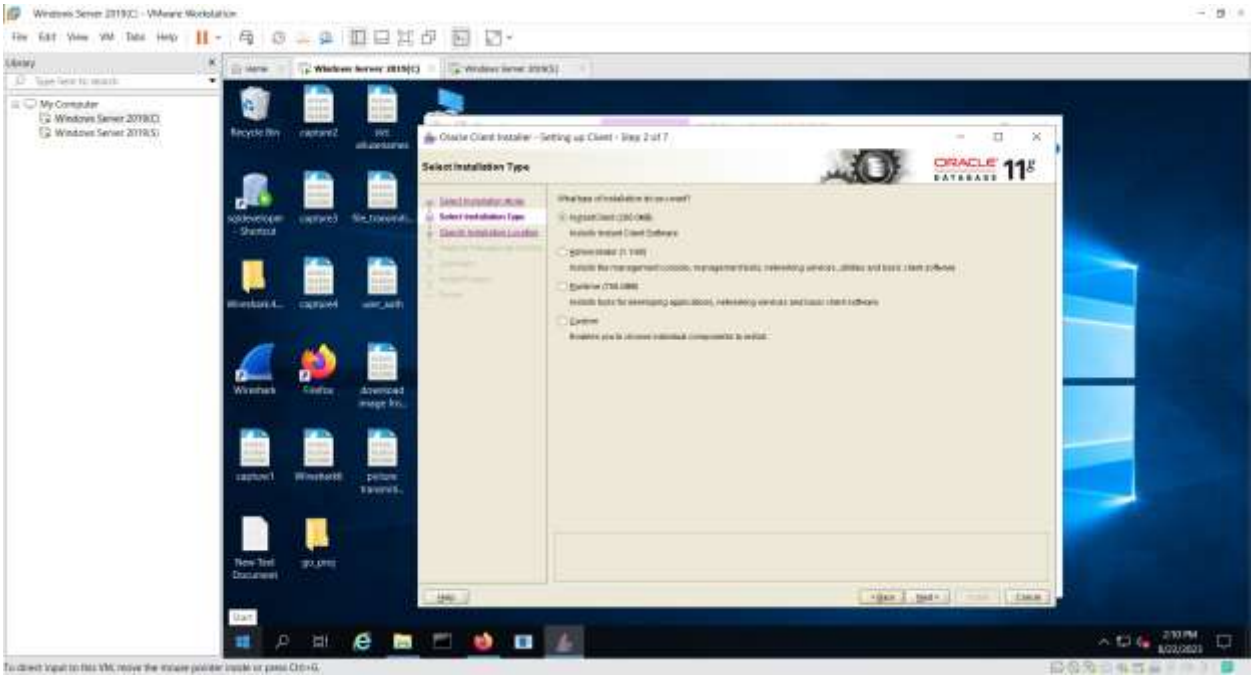
۵- بعد از تایید رمز، مراحل بعدی قابل گذر می‌باشد.

۲-۵- نصب اوراکل کلاینت

نصب اوراکل‌کلاینت تقریبا مشابه اوراکل سرور می‌باشد. اوراکل کلاینت از لینک زیر قابل دانلود است.

https://www.oracle.com/cis/database/technologies/instant-client/downloads.html

تنها نکته‌ی قابل اشاره درمورد نصب این نرم‌افزار این است که در فرایند نصب حتما گزینه‌ی Instantclient انتخاب شود.



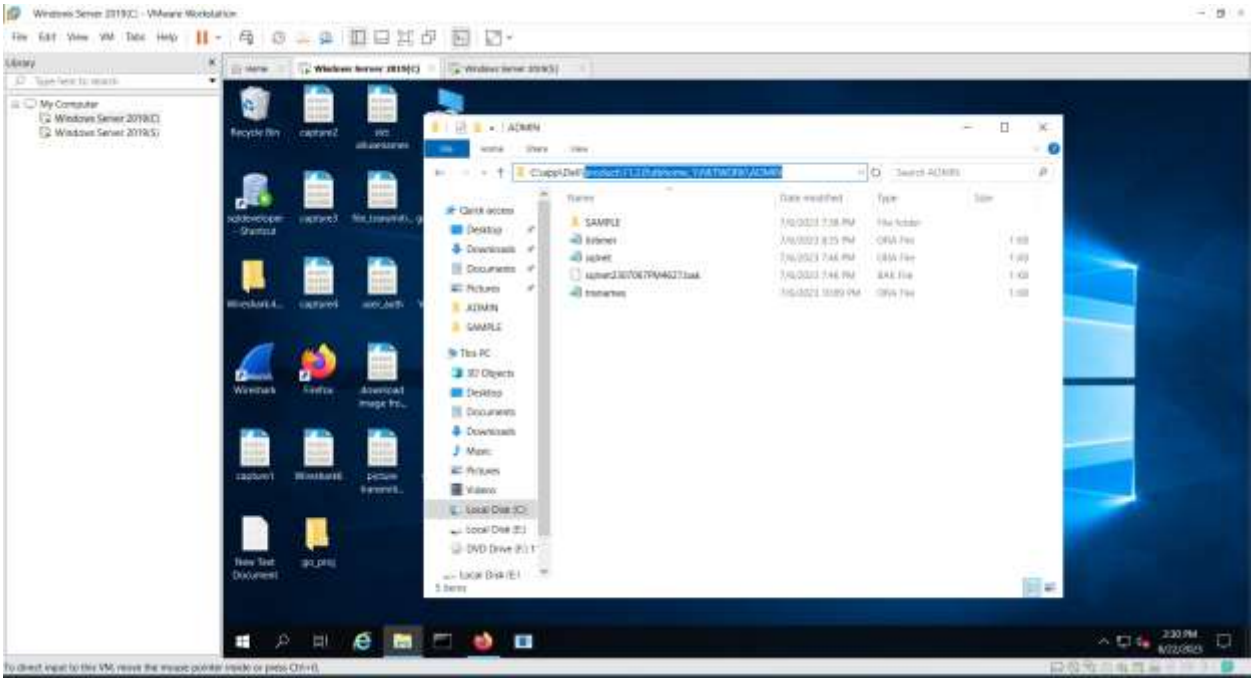
شکل (۲-۱۱)

چنانچه به هر دلیلی در دانلود این نرم‌افزار محدودیتی وجود داشته باشد، می‌توان اوراکل سرور را به جای اوراکل‌کلاینت بر روی ماشین مشتری نصب نمود.

۲-۵- تنظیم فایل tnsnames.ora

در قسمت‌های قبلی، نرم‌افزار اوراکل‌سرور بر روی ماشین سرویس‌دهنده و نرم‌افزار اوراکل‌کلاینت (یا اوراکل-سرور) بر روی ماشین مشتری نصب شد. همچنین در فصل قبل به نقش فایل tnsnames.ora در استخراج آدرس و پورت مربوط به نام پایگاه داده یا سرویس اشاره شد. حال باید این فایل برای برقراری اتصال تنظیم شود. بدیهی است که تمامی این عملیات‌ها در ماشین مشتری انجام می‌شود.(به فصل یک مراجعه شود.)

۱- ابتدا باید در مسیری که اوراکل‌کلاینت یا اوراکل سرور بر روی مشتری نصب شده است، وارد پوشه‌ی product\11.2.0\dbhome_1\NETWORK\ADMIN\... شد.

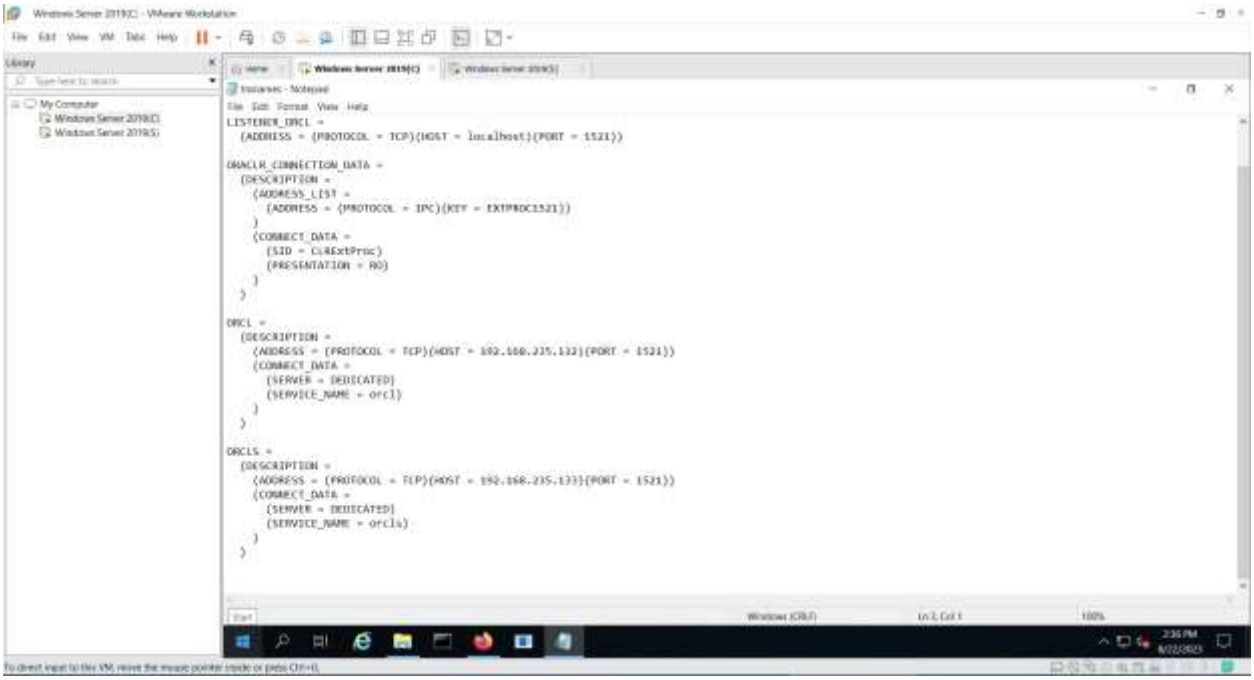


شکل (۲-۱۲): …\product\11.2.0\dbhome_1\NETWORK\ADMIN

۲- چنانچه فایل tnsnames.ora وجود نداشته باشد، به صورت دستی این فایل اضافه شود. اگر این فایل در Notepad باز شود، چنین اطلاعاتی مشاهده می‌شود. در این فایل می‌توان یک پایگاه داده جدید را نامگذاری و ایجاد کرد و آدرس و پورتی را به آن نسبت داد. به عنوان مثال در شکل زیر، پایگاه داده‌ی زیر اضافه شده است:

```
ORCLS =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.235.133)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = orcls)
    ))
```

این پایگاه داده orcls نام‌گذاری شده است. از آنجایی که ip ماشین سرویس‌دهنده برابر 192.168.235.133 است، پس همین مقدار برای متغیر Host تنظیم می‌شود. از آنجایی که درخواست‌ها به پورت 1521 ارسال می‌شود، پس مقدار آن برابر همین عدد قرار داده می‌شود.

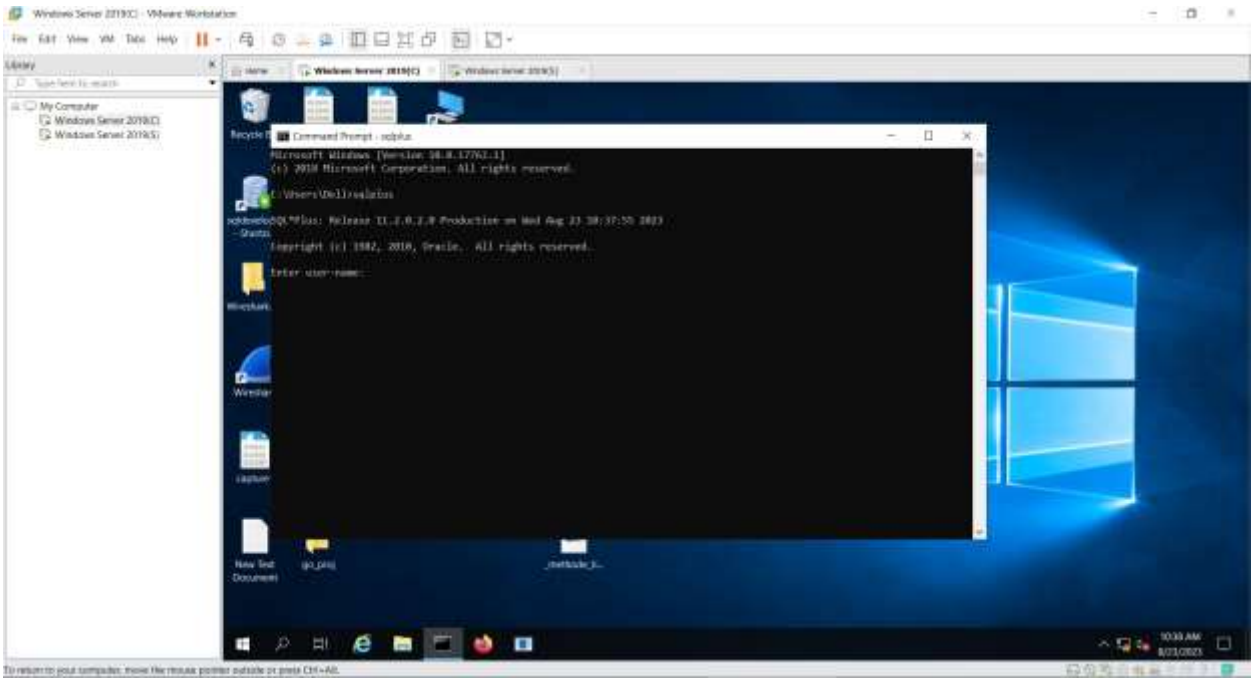


شکل (۲-۱۳): فایل tnsnames.ora

۲-۶- ایجاد اولین اتصال با SQLPlus

۱- برای انجام آزمایشات از دو محیط SQLPlus و Oracle server manager استفاده می‌شود. این موارد باید بر روی ماشین مشتری نصب شوند. کنسول SQLPlus به طور پیش‌فرض همراه اوراکل‌سرور و اوراکل‌کلاینت نصب می‌شود.

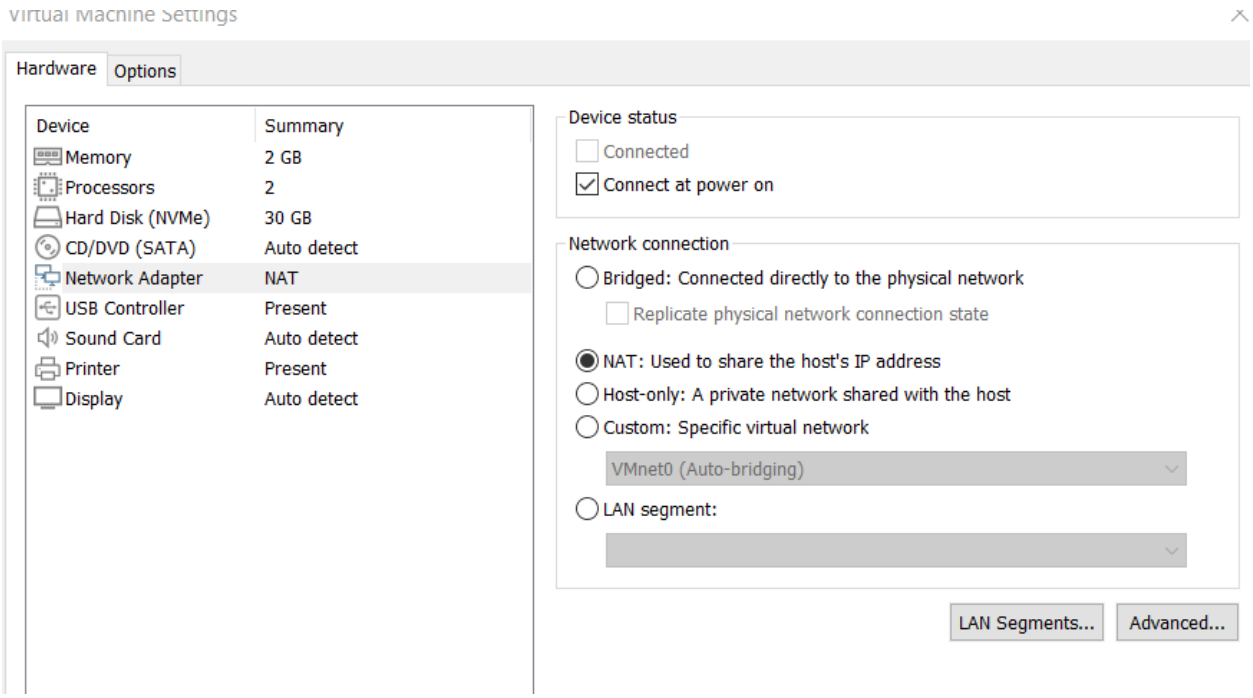
برای اطمینان از نصب SQLPlus بر روی سیستم مشتری، کافیست یک Command Promote باز شود و کلمه‌ی SQLPlus وارد شود. اکنون کنسول SQLPlus همراه با ورژن اوراکل نصب شده و پرسش نام‌کاربری نمایش داده می‌شود.



شکل (۲-۱٤): کنسول SQL Plus

۲- لازم است در سمت سرور، فایروال خاموش شود

۳- با دستور ipconfig در هر دو سمت سرور و مشتری، ip های دو ماشین استخراج شود و با دستور ping از اتصال دو ماشین به یکدیگر اطمینان حاصل کرد. در این راستا حتما تنظیمات شبکه‌ی دو ماشین، طبق شکل زیر منطبق باشد.



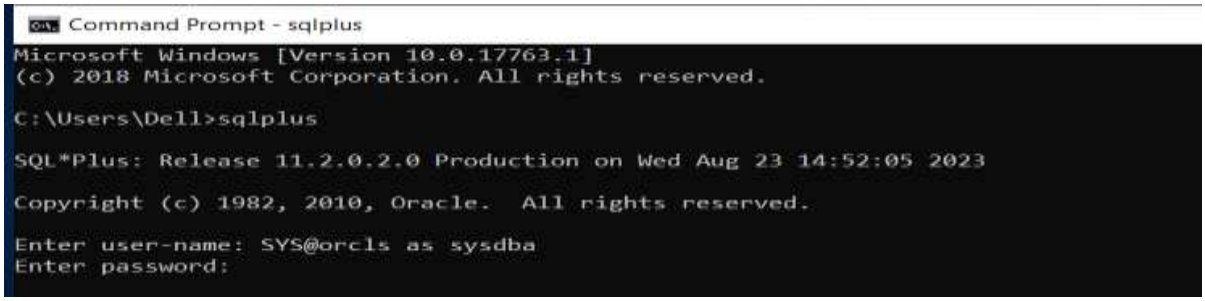
شکل (۲-۱۵): تنظیمات شبکه

٤- در این مرحله می‌بایست نام کاربری با قالب زیر نوشته شود (نوشتن حروف بزرگ و و کوچک تفاوتی ایجاد نمی‌کند.).

ورود مدیر سیستم : SYS@نام یا دامنه پایگاه داده as SYSDBA

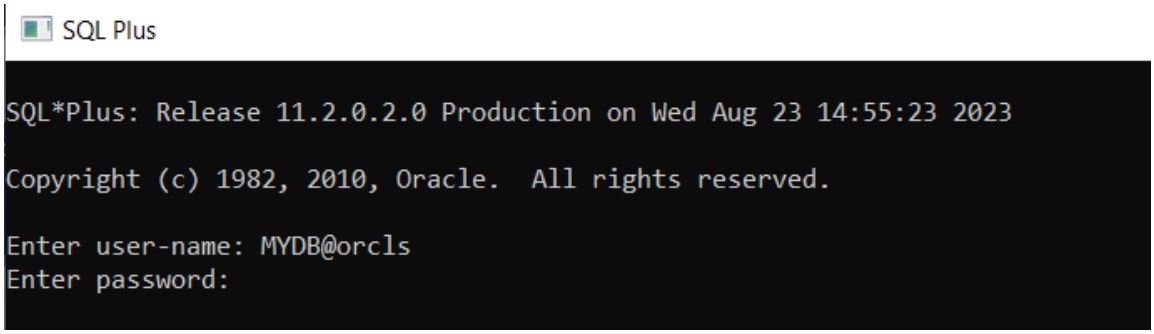
ورود کاربر عادی : نام‌کاربری@نام یا دامنه پایگاه داده

برای مثال در شکل زیر، مشاهده می‌شود که کاربر مدیر به پایگاه داده orcls متصل شده است. شکل (۲-۱۵)



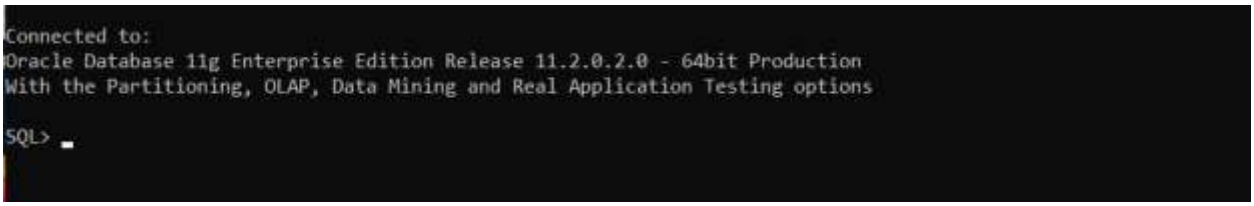
شکل (۲-۱۶)

و همچنین در شکل بعدی، کاربر عادی MYDB به orcls متصل شده است. شکل (۲-۱۶)



شکل (۲-۱۷)

۵- چنانچه پیغام زیر در کنسول نمایش داده شود، به این معناست که اتصال مشتری و پایگاه داده با موفقیت انجام شده است. در این صورت کاربر می‌تواند پرسش‌های sql را وارد کند. شکل (۲-۱۸)



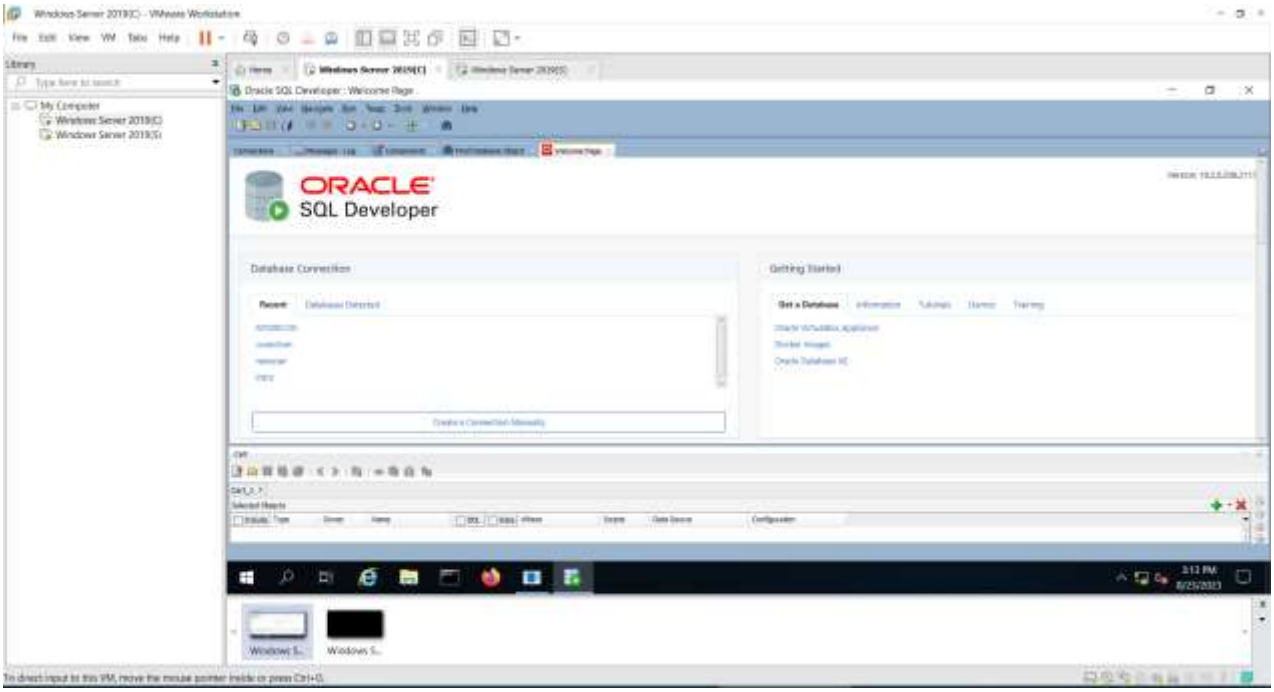
شکل (۲-۱۸): اتصال موفقیت‌آمیز

۲-۷- آشنایی و شروع کار با SQL developer

این نرم‌افزار یک نرم‌افزار نام‌آشنا برای برنامه‌نویسان پایگاه داده است. به کمک این نرم‌افزار، کاربر می‌تواند در نقش کاربر عادی یا مدیر وارد سیستم شود و مانند SQLPlus، درخواست‌های خود را بنویسد. به این طریق

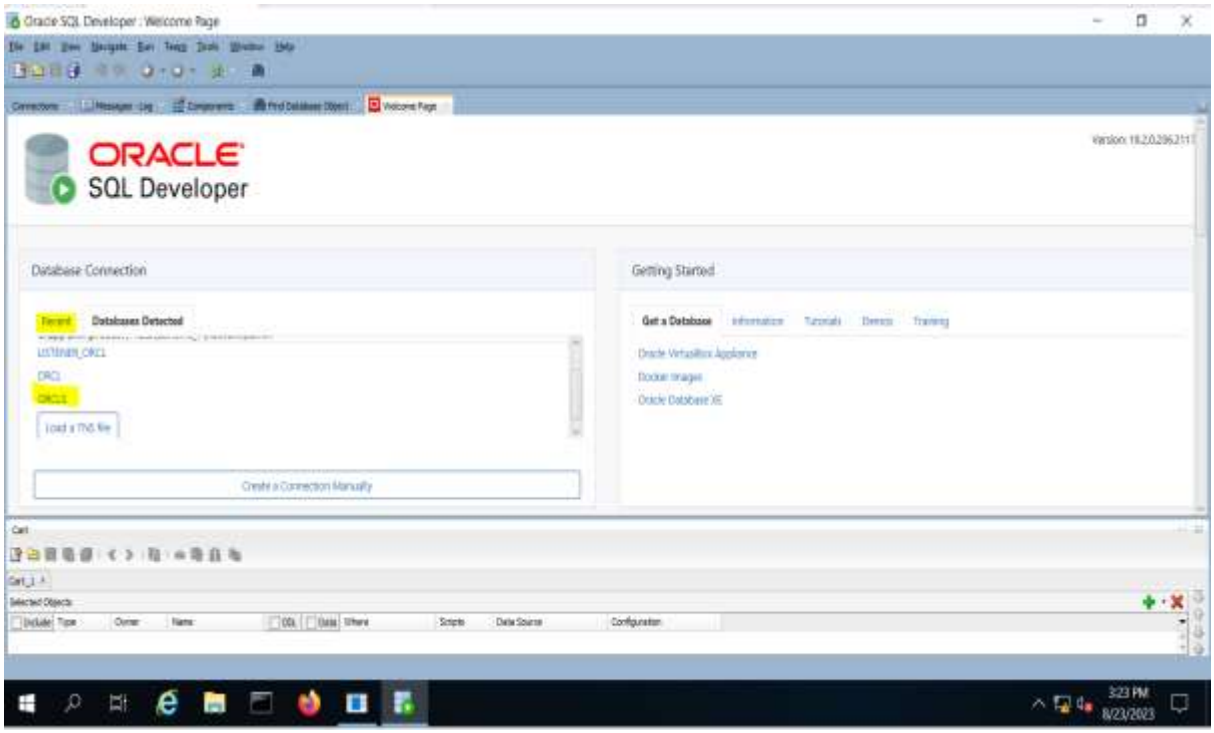
بسیاری از عملیات‌هایی که ممکن است با SQLPlus انجام شود، به صورت خودکار توسط این برنامه قابل انجام است. این نرم‌افزار می‌تواند به عنوان ابزاری کمکی بر روی مشتری نصب شود.

۱- از مهم ترین قابلیت‌های این نرم‌افزار، قابلیت ایجاد اتصال و تست اتصال است. همچنین به سادگی می‌توان به سیستم کاربر جدید اضافه کرد و دسترسی آن‌ها را مدیریت کرد. در اولین گام، SQL developer باز شود.



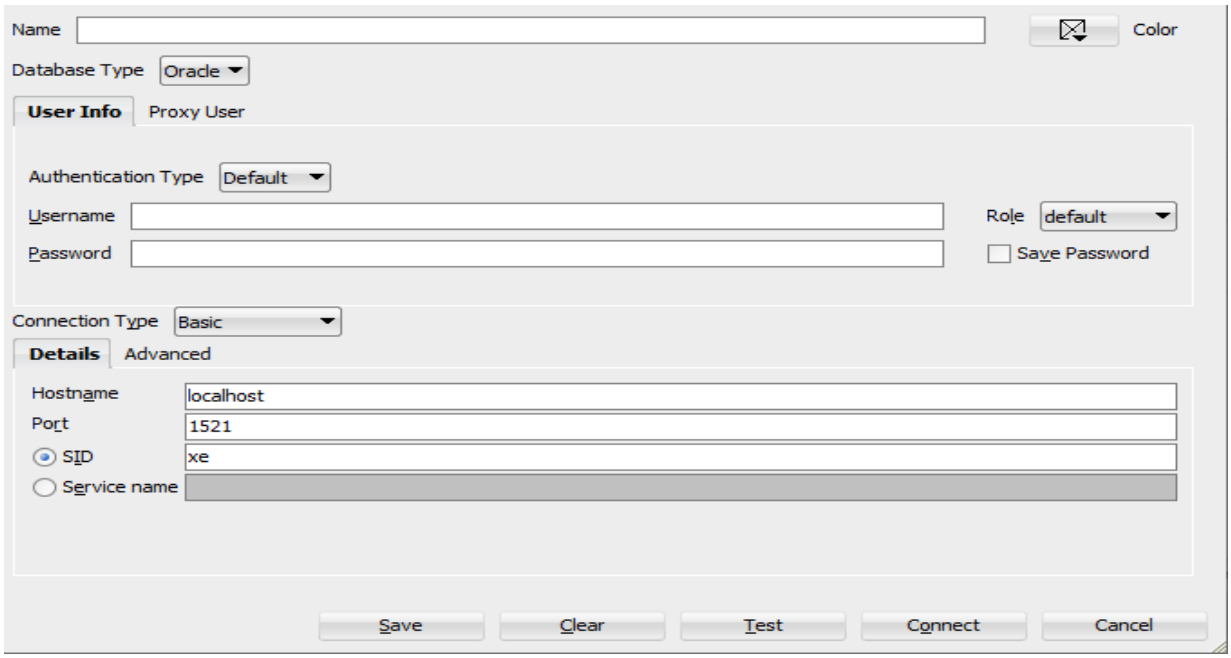
شکل (۲-۱۹): محیط SQLDeveloper

۲- چنانچه فایل tnsnames.ora به درستی تنظیم شده باشد، در قسمت Database detected لیست پایگاه داده‌های موجود، قابل مشاهده است. در شکل زیر مشاهده می‌شود که پایگاه داده‌ی orcls شناسایی شده است. در بخش recent می‌توان اتصالات اخیر را مشاهده کرد و به سرعت به آن وصل شد.(شکل۲-۲۰)



شکل (۲-۲۰): اتصال‌های موجود

۳- با کلیک بر روی گزینه‌ی Create connection Manually صفحه‌ی زیر نمایش داده می‌شود.



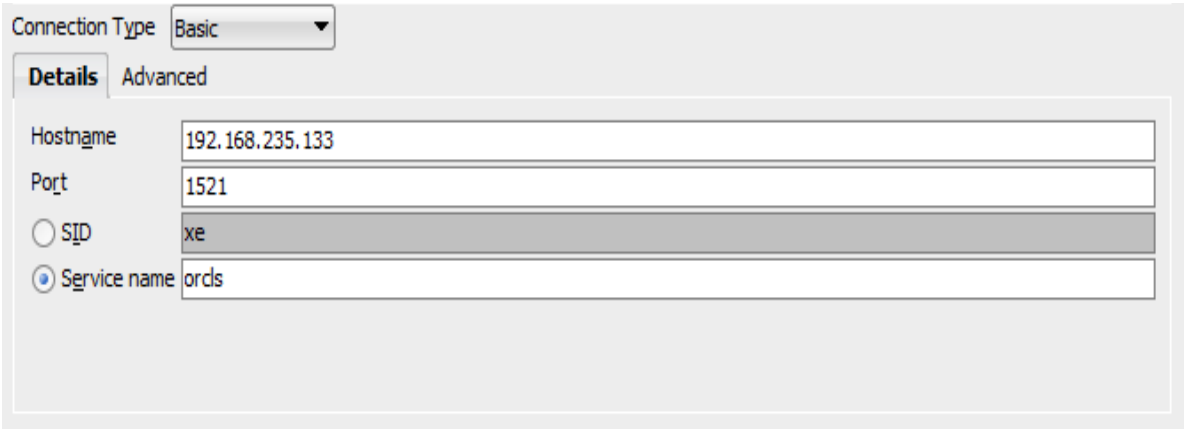
شکل (۲-۲۱): صفحه ایجاد اتصال جدید

٤- در بخش اول نام اتصال تعیین می‌شود. سپس نام کاربری و رمزعبور وارد می‌شود. درصورتی که کاربر به عنوان مدیر وارد می‌شود، باید در بخش Role نقش SYSDBA را انتخاب کند و در بخش نام کاربری، کلمه‌ی SYS را بنویسد. در غیر این صورت باید مقدار default داشته باشد.

شکل (۲-۲۲): ورود به عنوان کاربر مدیر

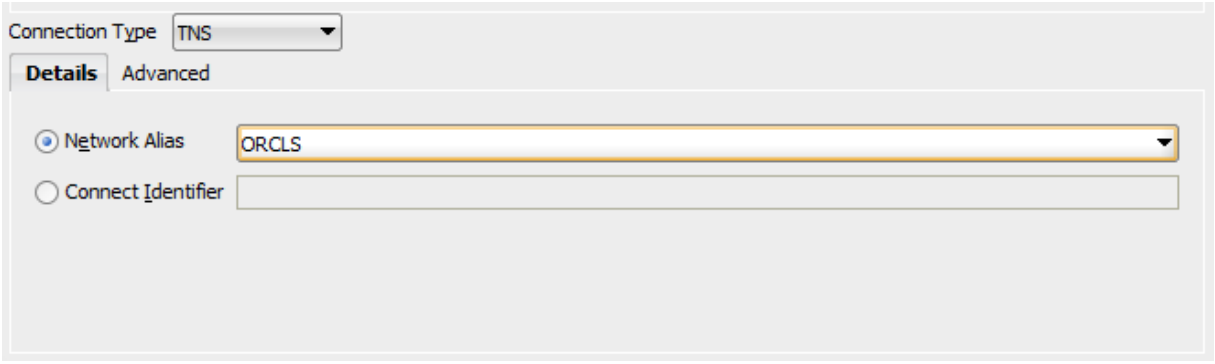
۵- در ادامه روش‌های مختلفی برای برقراری اتصال وجود دارد. دو روش آن یعنی Basic و TNS توضیح داده می‌شود:

- در روش Basic مانند شکل زیر گزینه‌ی Basic برای نوع اتصال انتخاب می‌شود و آدرس ، شماره‌ی پورت سرویس‌دهنده و نام سرویس وارد می‌شود.



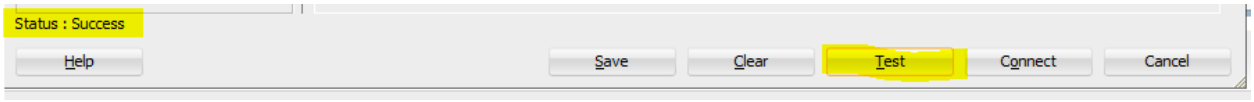
شکل (۲-۲۳): روش اتصال Basic

- در روش دوم نوع اتصال بر روی TNS تنظیم می‌شود و نام پایگاه داده ذکر می‌شود. پروتکل TNS، نام پایگاه داده را در فایل tnsnames.ora جست‌وجو می‌کند و اطلاعات آدرس و پورت پایگاه داده را به دست می‌آورد.



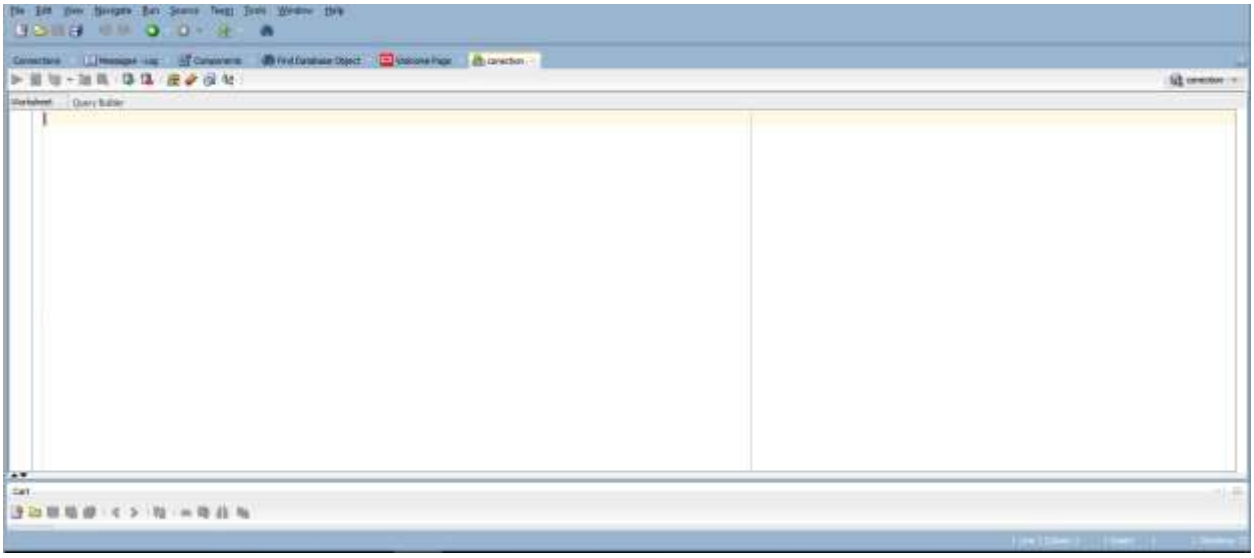
شکل (۲-۲٤): روش اتصال TNS

٦- در پایین منو، اتصال مشتری با پایگاه داده تست می‌شود.



شکل (۲-۲۳): اتصال موفقیت آمیز

۷- درصورت موفقیت‌آمیز بودن تست، گزینه‌ی اتصال انتخاب می‌شود و وارد کدنویسی SQL نمایان می‌شود.



شکل (۲-۲۵): محیط کدنویسی SQLDev

۸- چنان‌چه مجددا محیط SQL developer باز شود، می‌توان اتصال ایجادشده‌ی اخیر را در بخش recent مشاهده کرد و بار دیگر به آن وصل شد.

به این ترتیب در این فصل عملیات پیاده‌سازی، آماده‌سازی و اتصال یک ماشین مشتری به یک ماشین سرویس-دهنده‌ی اوراکل با موفقیت انجام شد.

فصل سوم

بررسی میدانی پروتکل TNS

۳-۱- مقدمه

در فصل قبل دو ماشین در نقش مشتری و سرور راه‌اندازی شد و نرم‌افزارهای لازم بر روی هر دو ماشین نصب گشت. پس از آن، تنظیمات مربوط به اتصال انجام شد و ادامه‌ی مراحل با ایجاد و تست اتصال به اتمام رسید. اکنون همه‌چیز مهیای بررسی میدانی پروتکل TNS می‌باشد. در این فصل عملکرد TNS در سناریوهای مختلف، برپایی و اتمام نشست و احراز هویت بررسی می‌شود. همچنین به تحلیل اطلاعات هر بسته پرداخته می‌شود.

۳-۲- معرفی و آشنایی با نرم‌افزار Wireshark

نرم‌افزارهای ردیابی مانند Wireshak نرم‌افزارهایی هستند که به مهندسان در بررسی پروتکل‌ها و اتصالات بین ماشین‌ها بسیار کمک می‌کند. این نرم‌افزارها به اتصالات بین دو ماشین گوش فرا می‌دهند و بسته‌های جابه‌جا شده میان آن‌ها را نمایش می‌دهند.

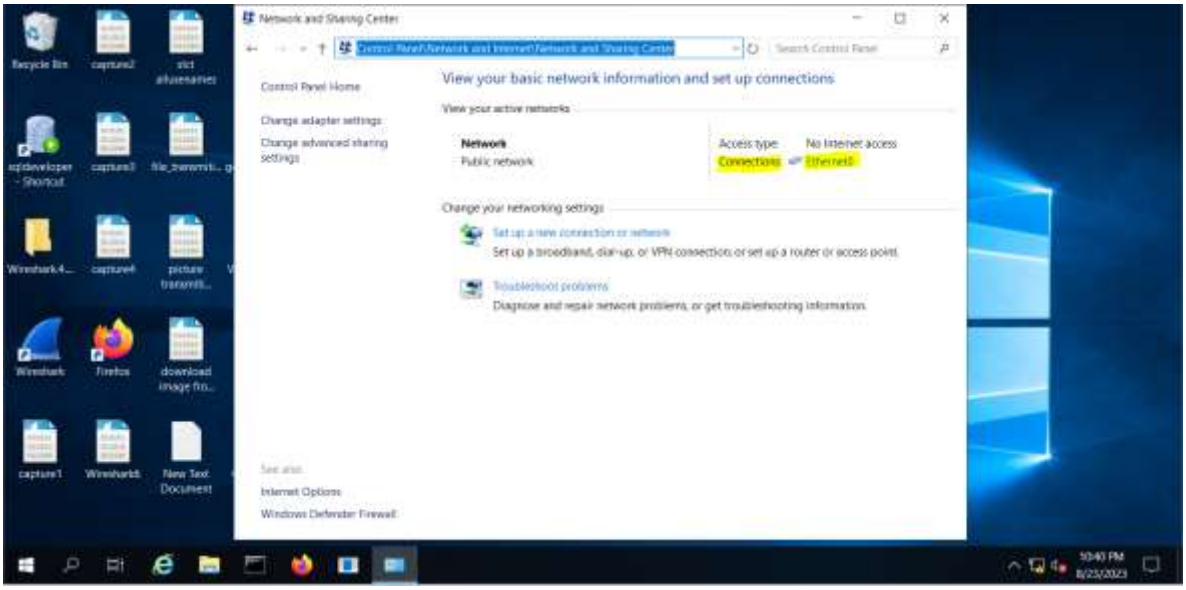
از امکانات مفید این نرم‌افزارها، تنوع استفاده از فیلترها و بررسی ساختار و معماری هر بسته می‌باشد. نرم‌افزاری که برای تحلیل این پروتکل استفاده می‌شود، نرم‌افزار کارآمد Wireshark می‌باشد. این نرم‌افزار می‌بایست بر روی ماشین مشتری نصب شود تا اطلاعاتی که در قالب بسته‌های TNS یا TCP ارسال می‌شود را ردیابی کند. با بررسی این بسته‌ها سعی می‌شود تا حد ممکن هم به رفتار و عملکرد پروتکل و هم به نوع و محتوای بسته‌های جابه‌جا شده پی برد.

در ادامه محیط نرم‌افزار معرفی می‌شود. تمامی مراحل زیر در ماشین مشتری انجام می‌شود:

۱- قبل از هر چیز می‌بایست مسیر زیر را وارد کرد و به صفحه‌ی Networksharingcenter وارد شد.

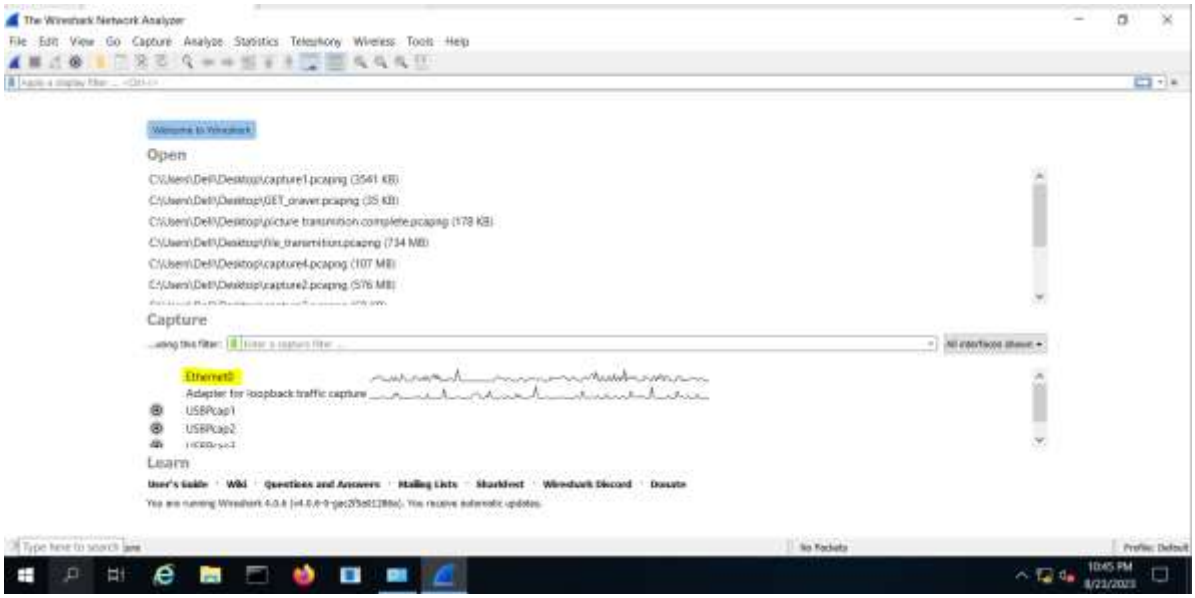
Control Panel > Network and Internet > Network and Sharing Center

۲- همانطور که قابل مشاهده است، در بخش اتصالات، تنها اتصالی که به شبکه وصل است Ethernet0 نام دارد.



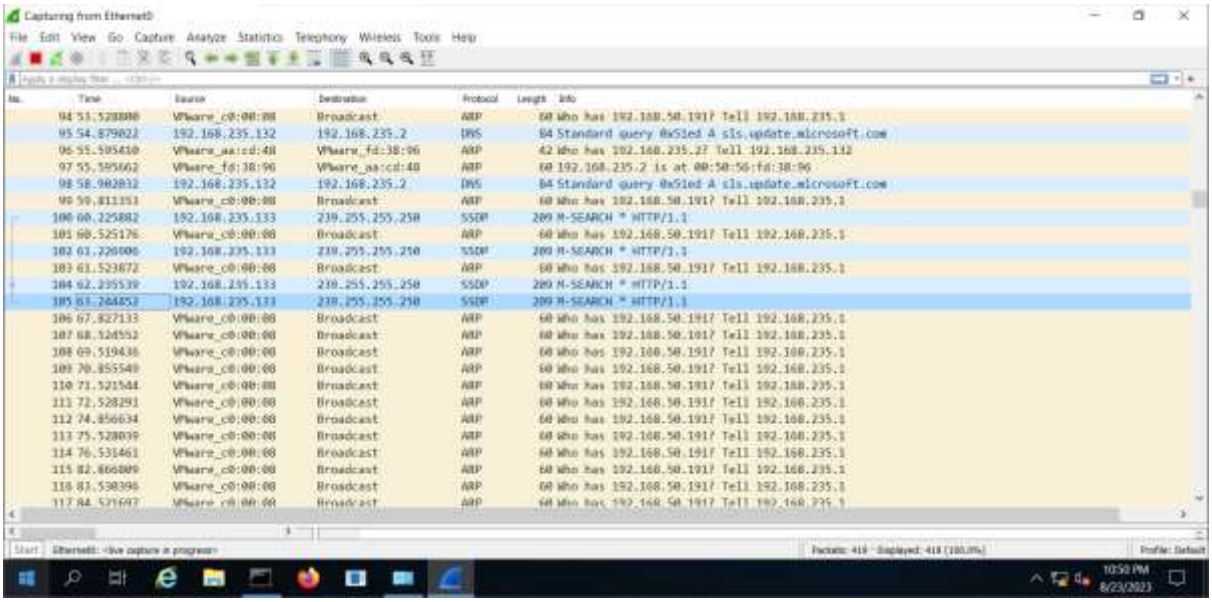
شکل (۳-۱): بررسی نام اتصال

۳- حال باید وارد نرم‌افزار wireshark شد و اتصال Ethernet0 را برای ردیابی انتخاب کرد.



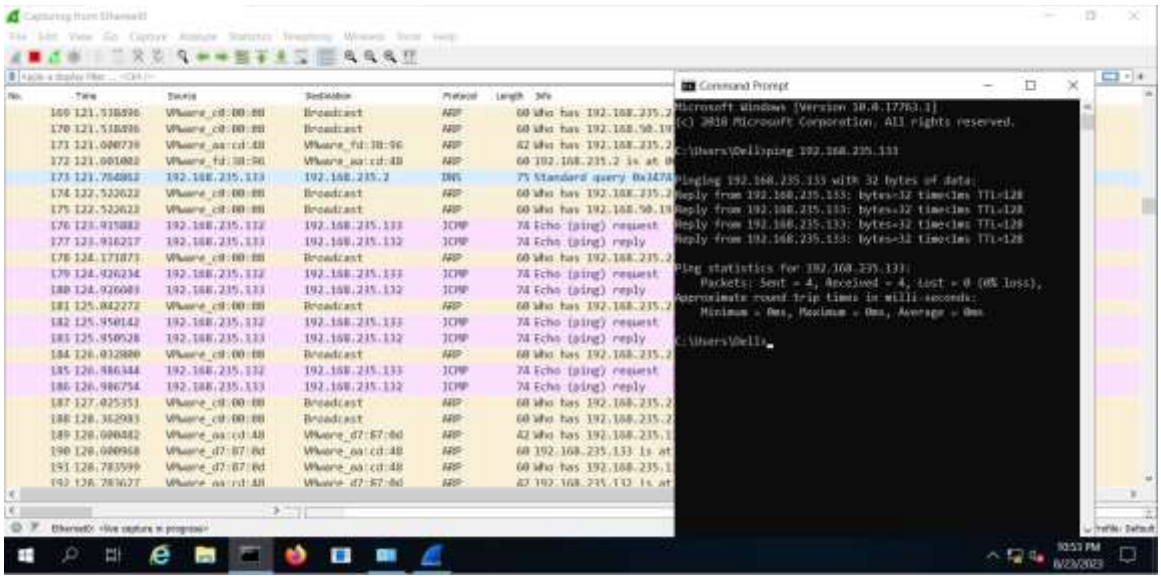
شکل (۳-۲): انتخاب Ethernet0

٤- پس از انتخاب اتصال، تمامی بسته‌های جابه‌جا شده در این اتصال با ذکر زمان، نوع پروتکل، مبدا، مقصد، طول بسته و اطلاعات اضافه قابل نمایش است.



شکل (۳-۳): تمام بسته‌های Ethernet0

۵- اکنون بهتر است مشتری، سرویس‌دهنده را Ping کند تا بررسی شود اتصال درست انتخاب شده است یا خیر. در شکل زیر مشاهده می‌شود که بسته‌ها (در قالب سطرهای صورتی) به درستی در wireshark نمایش داده شده است.



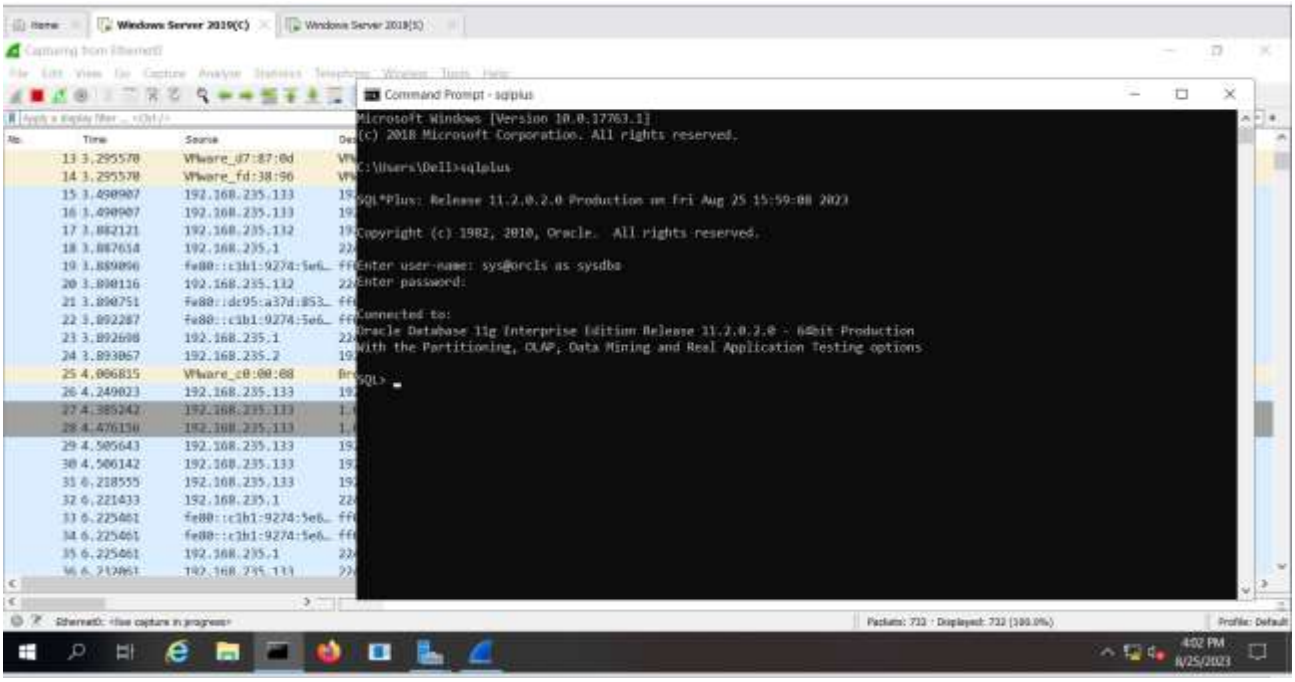
شکل (۳-٤): تست اتصال به سرویس‌دهنده

۳-۳- تحلیل سناریوها

پس از اطمینان از عملکرد صحیح نرم‌افزار wireshark، باید سلسله عملیات‌هایی بین مشتری و سرویس‌دهنده انجام داد تا از عملکرد پروتکل TNS آگاه شد. در این راستا سعی شده است که چند سناریوی مهم، به صورت کلی بررسی و تحلیل شود. جزییات بسته‌های ردوبدل شده در قسمت‌های بعد گزارش داده می‌شود.

۳-۳-۱- اتصال کاربر به سیستم

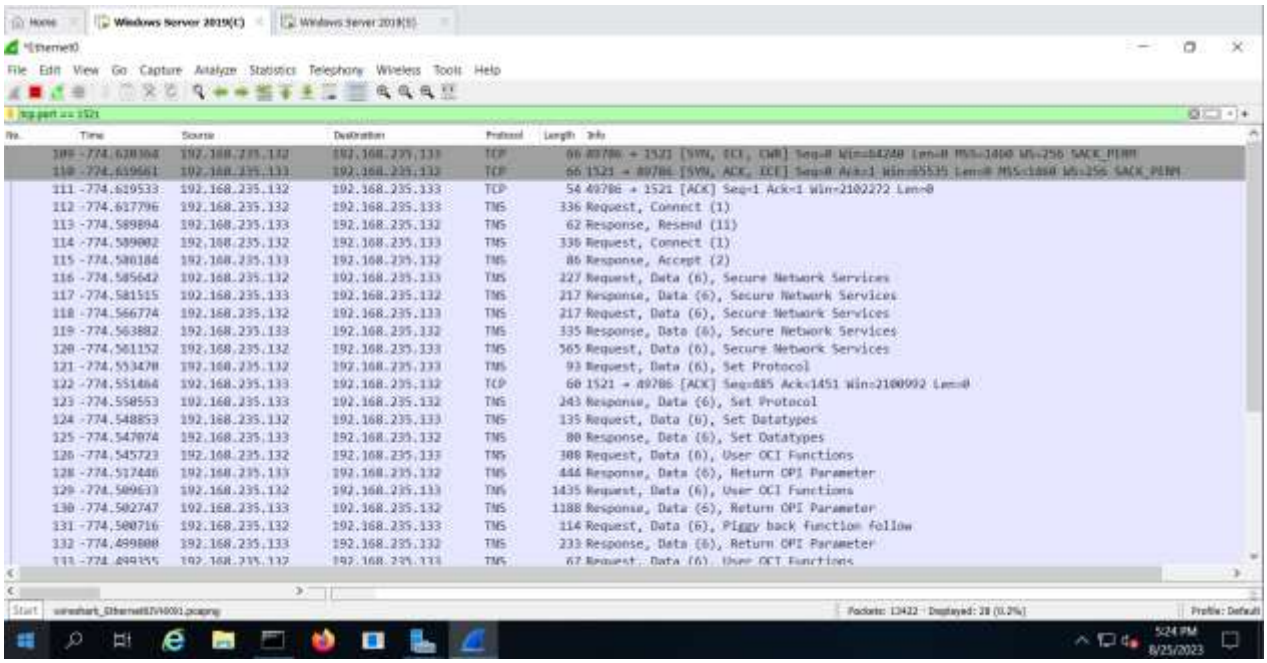
۱- نرم‌افزار wireshark به گونه‌ای تنظیم می‌شود که شنونده‌ی اتصال Ethernet0 باشد. همزمان از سمت مشتری، باید وارد صفحه‌ی SQLplus شد. طبق روال گذشته، نام کاربری و رمزعبور مدیر وارد می‌شود. شکل (۳-۵)



شکل (۳-۵)

اکنون اتصال با موفقیت برقرار شده است.

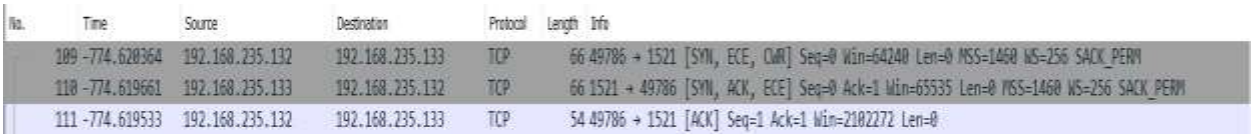
۲- در قدم بعد باید وارد صفحه‌ی wireshark شد. در بالای صفحه کادری وجود دارد که می‌توان با کمک مجموعه‌ای از عبارات شرطی، بسته‌های مورد جست‌وجو را استخراج کرد. از آنجا که درخواست‌ها به پورت ۱۵۲۱ ارسال شده است، پس عبارت tcp.port == 1521 در کادر فیلتر بالا نوشته می‌شود.



شکل (۳-٦): بسته‌های با پورت مقصد ۱۵۲۱

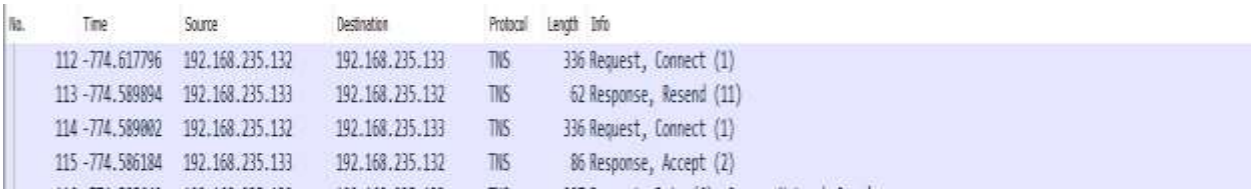
۳- همانطور که مشاهده می‌شود، تمامی بسته‌هایی که به پورت ۱۵۲۱ ارسال شده است، در قالب بسته‌های TCP و TNS تبادل شده است.

٤- در ابتدا مشتری دو بسته‌ی TCP را برای بررسی صحت اتصال پورت ۱۵۲۱، به سرویس دهنده ارسال می‌کند. سپس سرویس‌دهنده یک بسته‌ی Ack برای تایید اتصال به مشتری ارسال می‌کند.



شکل (۳-۷): بسته‌های صحت اتصال

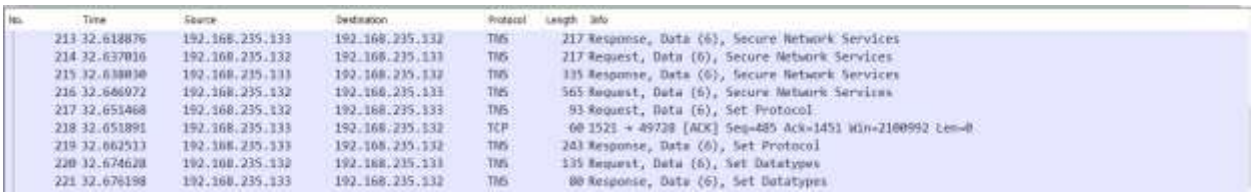
۵- در گام بعد، مشتری یک بسته‌ی TNS با نوع شماره‌ی ۱ (Connect) به سرور ارسال می‌کند. در صورت عدم دریافت بسته‌ی Connect، سرویس‌دهنده با ارسال بسته‌ی شماره‌ی ۱۱ (Resend) به مشتری می‌فهماند که باید بسته مجددا ارسال شود. در بسته‌ی Connect اطلاعات مهمی در رابطه با اتصال مشتری به سرویس دهنده ارسال می‌شود که به جزییات آن بعدا پرداخته می‌شود. در صورت تایید این اطلاعات توسط سرویس‌دهنده، بسته‌ی شماره‌ی ۲ (Accept) را به مشتری ارسال می‌کند.



No.	Time	Source	Destination	Protocol	Length	Info
112	-774.617796	192.168.235.132	192.168.235.133	TNS	336	Request, Connect (1)
113	-774.589894	192.168.235.133	192.168.235.132	TNS	62	Response, Resend (11)
114	-774.589002	192.168.235.132	192.168.235.133	TNS	336	Request, Connect (1)
115	-774.586184	192.168.235.133	192.168.235.132	TNS	86	Response, Accept (2)

شکل (۳-۸): فرآیند اتصال

۶- پس از این مرحله، بسته‌های شماره‌ی ۶ (Data) نقش پررنگی را در تبادل اطلاعات بر عهده می‌گیرد. برای شروع نشست و عملیات پرسش و پاسخ می‌بایست توافقاتی مانند تنظیم نوع داده‌ها، پروتکل‌های قابل استفاده و ارسال مشخصات سرویس‌های امنیتی مشتری، انجام شود.



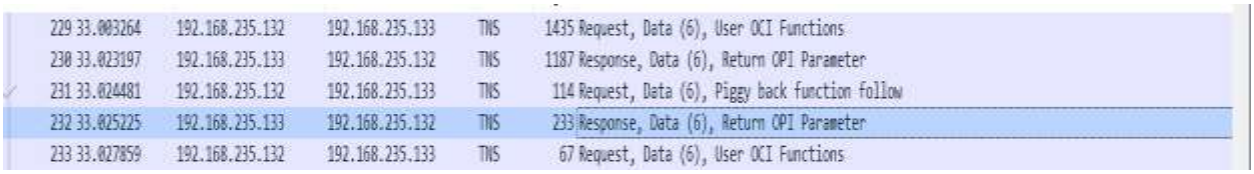
No.	Time	Source	Destination	Protocol	Length	Info
213	32.618876	192.168.235.133	192.168.235.132	TNS	217	Response, Data (6), Secure Network Services
214	32.637016	192.168.235.132	192.168.235.133	TNS	217	Request, Data (6), Secure Network Services
215	32.638030	192.168.235.133	192.168.235.132	TNS	135	Response, Data (6), Secure Network Services
216	32.646972	192.168.235.132	192.168.235.133	TNS	565	Request, Data (6), Secure Network Services
217	32.651468	192.168.235.132	192.168.235.133	TNS	93	Request, Data (6), Set Protocol
218	32.651891	192.168.235.133	192.168.235.132	TCP	60	1521 → 49720 [ACK] Seq=485 Ack=1451 Win=2100992 Len=0
219	32.662513	192.168.235.133	192.168.235.132	TNS	243	Response, Data (6), Set Protocol
220	32.674628	192.168.235.132	192.168.235.133	TNS	135	Request, Data (6), Set Datatypes
221	32.676198	192.168.235.133	192.168.235.132	TNS	80	Response, Data (6), Set Datatypes

شکل (۳-۹): توافقات اولیه‌ی مشتری و سرویس‌دهنده

۷- پس از انجام توافقات میان مشتری و سرویس‌دهنده، زمان انجام احراز هویت فرا می‌رسد. ابتدا مشتری طی بسته‌ای از سرور درخواست کلید نشست را می‌کند. کلید نشست حاوی اطلاعات ارزشمندی درباره‌ی نشست و سرویس درخواست‌شده است. این کلید توسط سرویس‌دهنده به مشتری ارسال می‌شود. تمامی این بسته‌ها از نوع بسته‌ی شماره ۶ (Data) است.

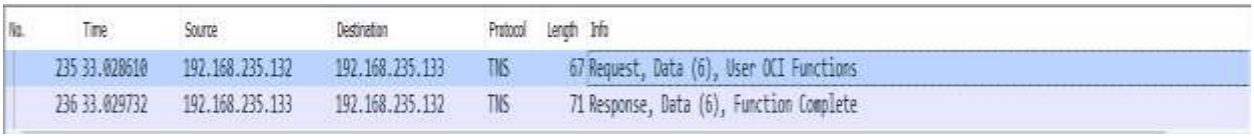
۸- سپس مشتری بسته‌ای شامل اطلاعات احراز هویت را در قالب بسته‌ی Data ارسال می‌کند و سرویس‌دهنده، دسترسی‌ها و پارامترهای هویتی مربوط به کاربر را در قالب یک بسته ارسال می‌کند و به این طریق عملیات ورود کاربر به اتمام می‌رسد.

نکته: در طول تمام عملیات‌ها، درخواست‌های کاربر با عنوان User OCI functions یا Piggy back functions follow و پاسخ‌های کاربر با عنوان Return OPI Parameters به مشتری ارسال می‌شوند.



شکل (۳-۱۰): بسته‌ی پاسخ OPI

۹- پس از اتمام درخواست احراز هویت، مشتری بسته‌ای نوع Data جهت تایید روند درخواست، ارسال می‌کند. به محض دریافت این پیام، سرویس‌دهنده در پاسخ بسته‌ای به نشانه‌ی ختم فرایند احراز هویت به مشتری ارسال می‌کند.



شکل (۳-۱۱): بسته‌ی پایان عملیات

۳-۳-۲- درخواست به پایگاه داده

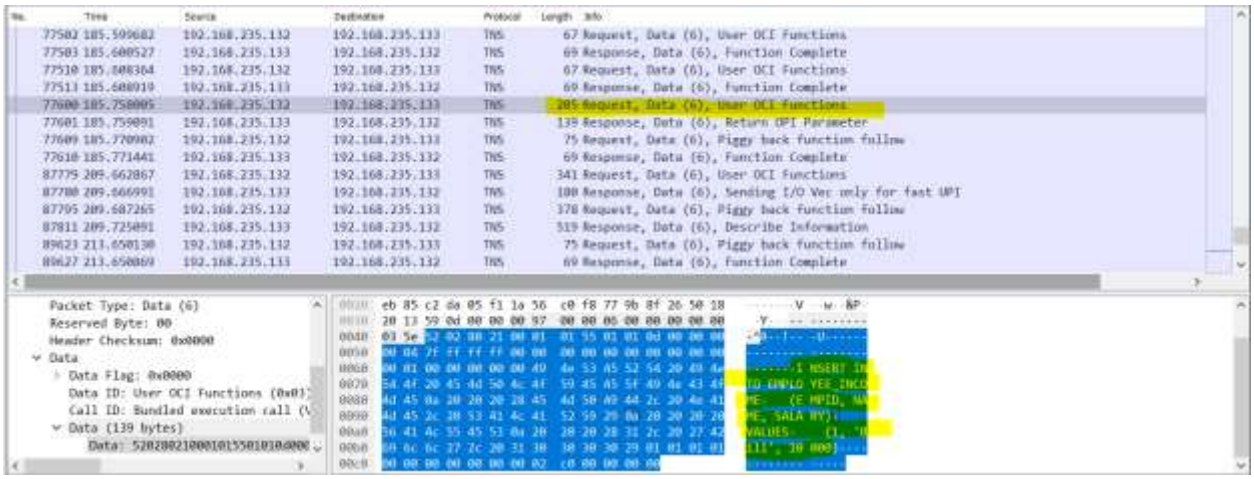
۱- برای جلوگیری از شلوغی اطلاعات Wireshark، عملیات استخراج بسته‌ها از زمان بعد احراز هویت، آغاز شود. برای انجام این آزمایش، از قبل یک جدول ساده با عنوان Employee_income ایجاد شود.

۲- حال باید عملیات ردیابی توسط Wireshark فعال شود.

۳- اکنون دستور insert زیر برای افزودن یک سطر به جدول وارد می‌شود.

Insert into Employee_income (Empid, Name,Sallary) Values (1, ‘Bill’, 1000);

٤- با جست‌وجو در میان بسته‌ها می‌توان متوجه شد که این درخواست در غالب یک بسته‌ی نوع Data به سرویس‌دهنده ارسال شده است. این دستور با عنوان User OCI Functions ذکر می‌شود

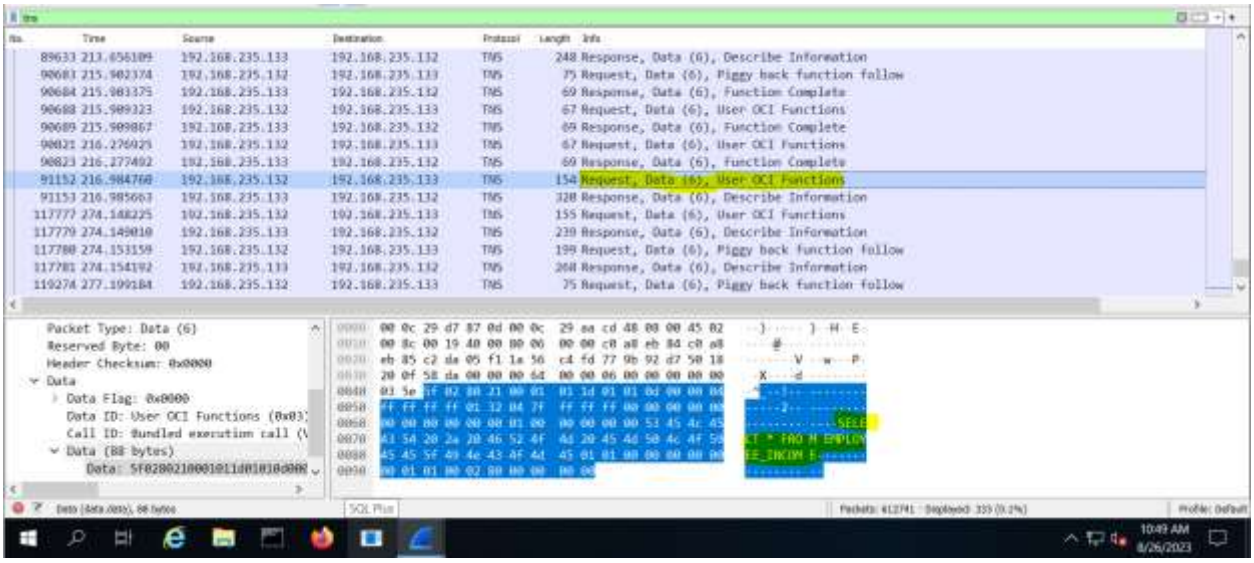


شکل (۳-۱۲): بسته‌ی پرسش مشتری حاوی متن پرسش

۵- اکنون دستور Select مورد بررسی قرار می‌گیرد.

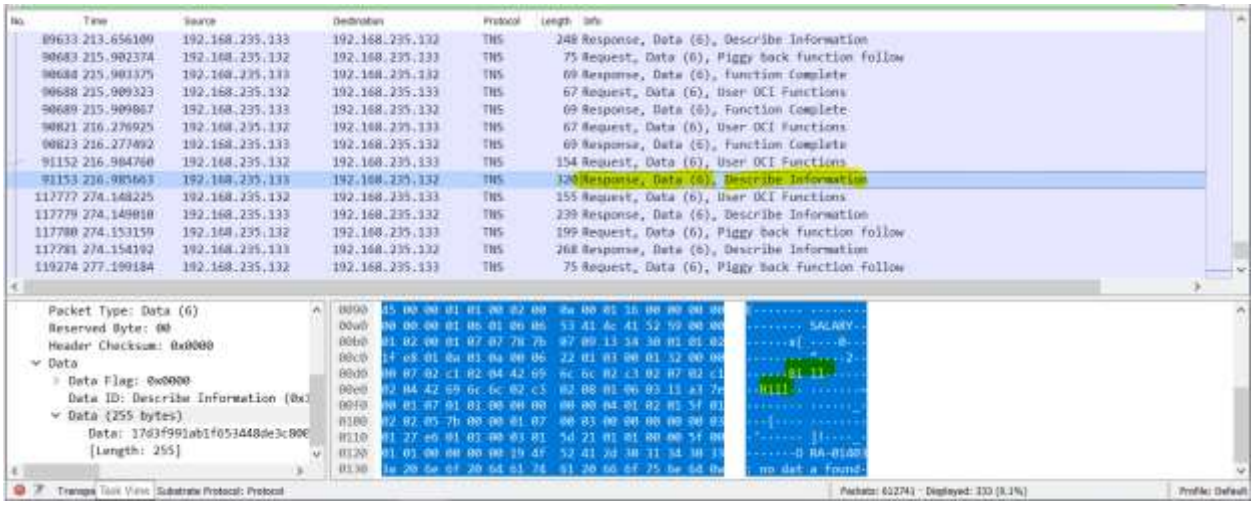
```
Select  * from Employee_income;
```

٦- درخواست بالا مجدد در قالب یک بسته‌ی Data به سرویس دهنده ارسال شده است.



شکل (۳-۱۳): بسته‌ی پرسش مشتری حاوی متن پرسش

۷- پاسخ‌های سرویس‌دهنده در قالب بسته‌های نوع Data با عنوان Discribe information به مشتری ارسال می‌گردد.

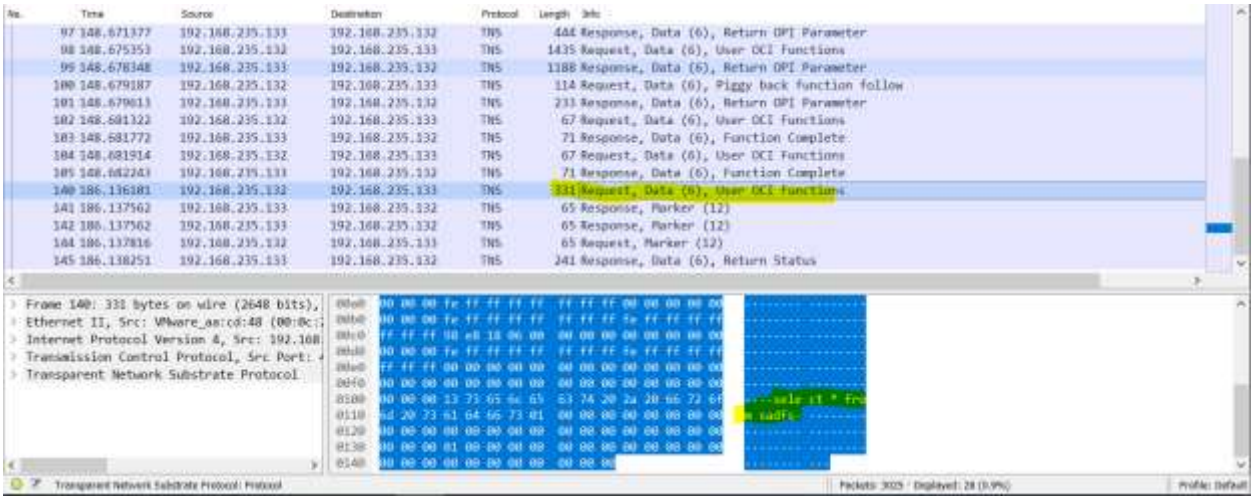


شکل (۳-۱٤): بسته‌ی پاسخ پرسش حاوی داده‌ی "Bill"

۸- اکنون درخواستی از سرویس‌دهنده می‌شود که سرویس‌دهنده پاسخی برای آن ندارد. زیرا جدولی به چنین نامی ایجاد نشده است.

```
Select * from sadfs;
```

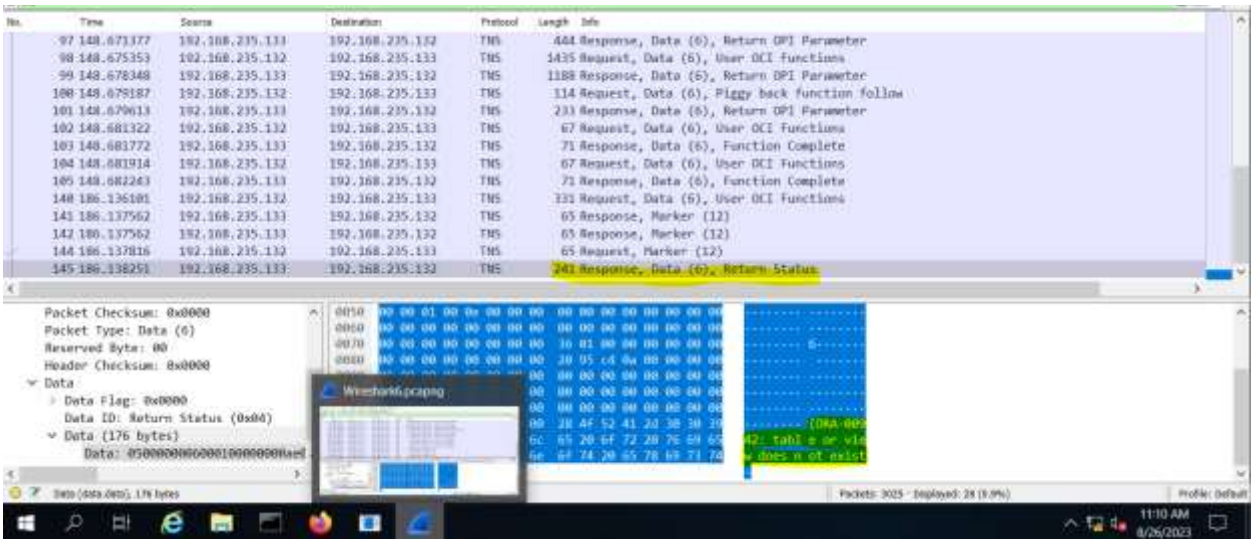
۹- ابتدا درخواست به سرویس‌دهنده ارسال و شناسایی می‌شود.



شکل (۳-۱۵): بسته‌ی پرسش نامعتبر

۱۰- سرویس‌دهنده در فرصتی‌که دارد نمی‌تواند پاسخ را پیدا کند. بنابراین برای جست‌وجوی بیشتر، بسته‌های نوع ۱۲ (Marker) را ارسال می‌کند. بسته‌ای که همانند یک interrupt عمل می‌کند.

۱۱- در نهایت جست‌وجو با ارسال بسته‌ی Return status در قالب بسته‌ی Data به پایان می‌رسد. در این بسته پیغام عدم وجود جدول ذکر شده است.



شکل (۳-۱۶): بسته‌ی نتیجه پرسش

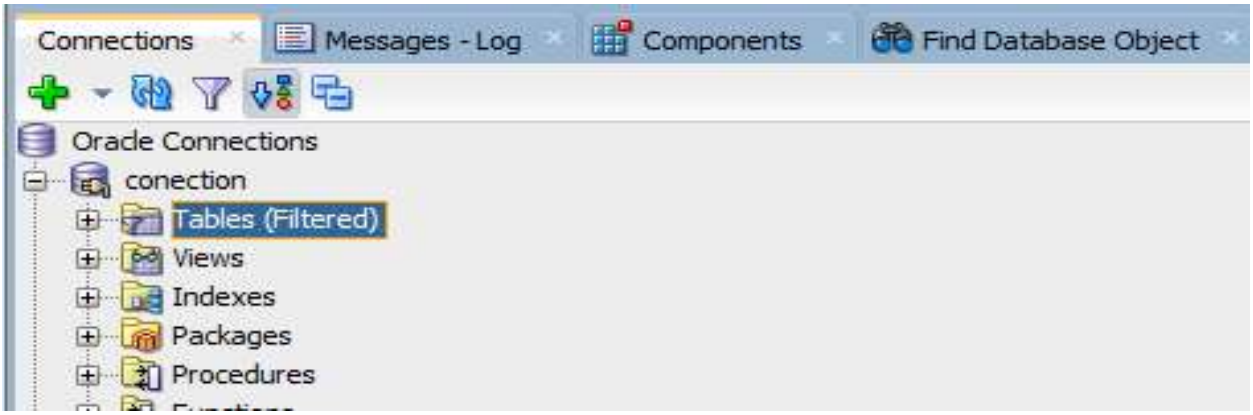
۳-۳-۳- ارسال داده‌های پرحجم

داده‌هایی که حجم زیادی دارند، با نوع BLOB ذخیره می‌شوند. برای ایجاد و ارسال این نوع داده‌ها بهتر است به جای محیط SQLPlus، در محیط SQL manager کدنویسی کرد. ابتدا یک جدول با نام Video و صفات زیر تشکیل شود.

SYS.VIDEO	
VID_ID	NUMBER
VID_NAME	VARCHAR2 (100 BYTE)
VIDEO	BLOB

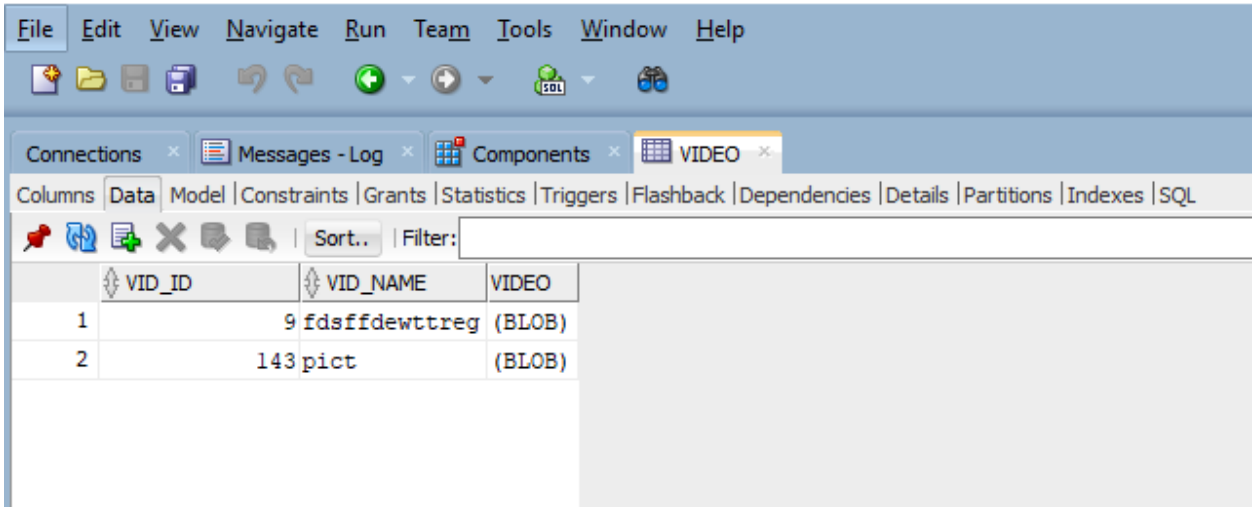
شکل (۳-۱۷): جدول ویدیو

۱- از قسمت Tab اتصالی که با آن ورود انجام شده است، بر روی گزینه‌ی جداول کلیک شود.



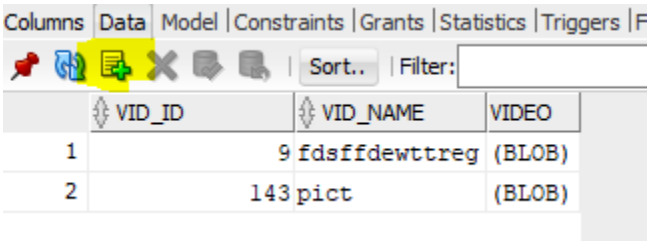
شکل (۳-۱۸): بخش جداول

۳-حال باید جدول Video انتخاب شود و وارد بخش Data شد



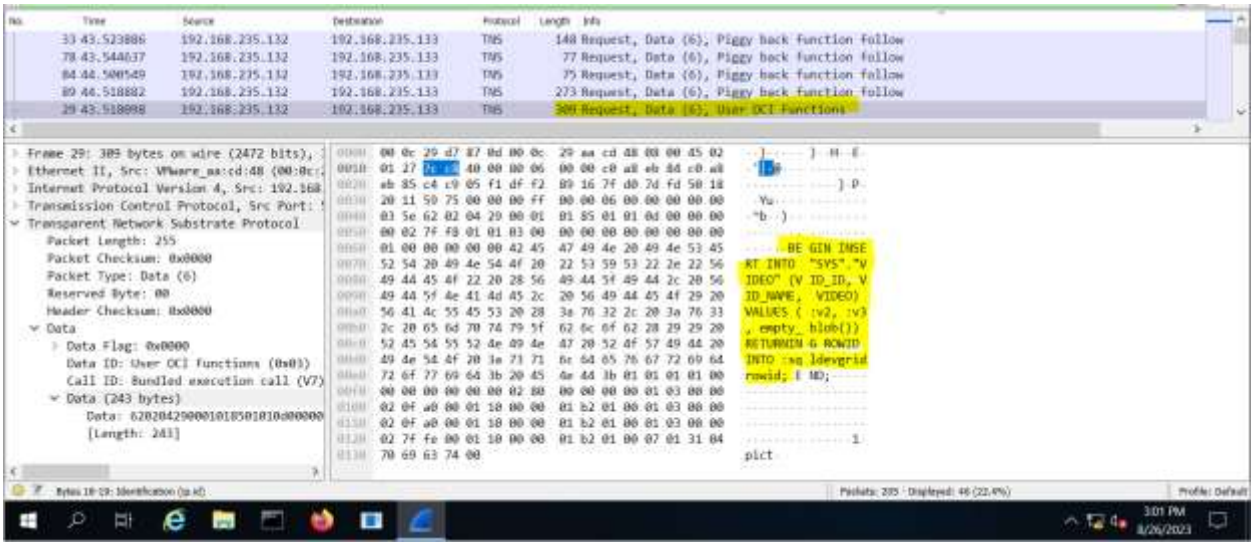
شکل (۳-۱۹): جدول ویدیو همراه با داده

٤- حال Wireshark باید ردیابی را آغاز نماید. بر روی گزینه‌ی شکل زیر(insert) کلیک کرده و یک سطر حاوی یک عکس، به سرویس‌دهنده ارسال می‌شود.



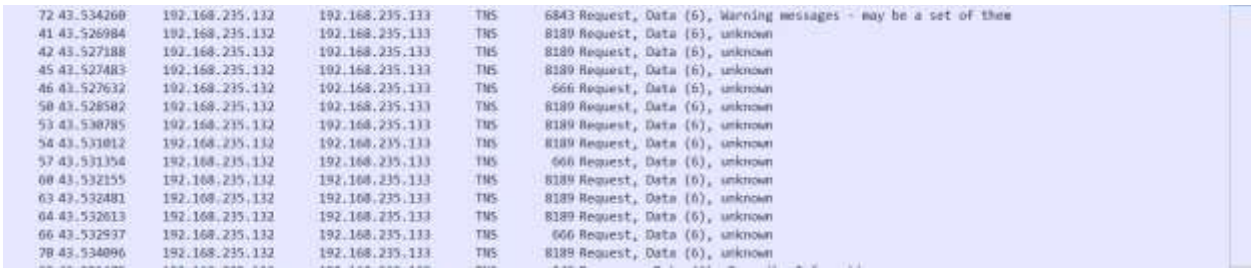
شکل (۳-۲۰): افزودن سطر داده

۵- پس از ارسال عکس، روی گزینه‌ی commit که کنار insert است کلیک شود. حال زمان بررسی wireshark است. مشاهده می‌شود که مشتری یک درخواست insert با کمک یک بسته‌ی OCI function به سرویس‌دهنده فرستاده است.



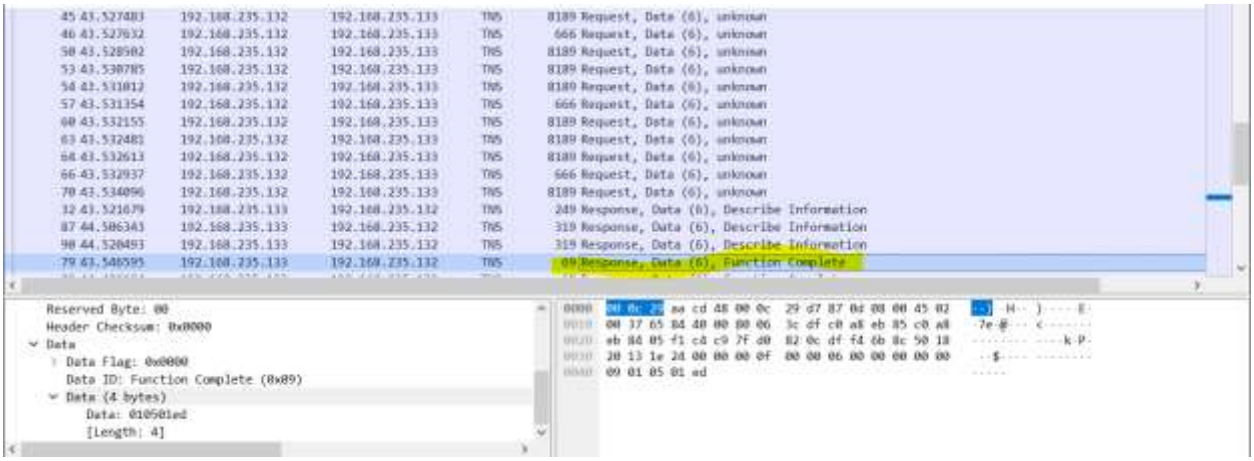
شکل (۳-۲۱): بسته‌ی حاوی پرسش افزودن سطر

۶- با ارسال این درخواست، فرآیند ارسال عکس از سمت مشتری در قالب چندین بسته‌ی Data آغاز می‌شود.



شکل (۳-۲۲): بسته‌های آپلود داده‌ی عکس از سوی مشتری

۷- با ارسال بسته‌ی function complete از نوع Data، عملیات ذخیره عکس از سوی سرویس‌دهنده، پایان‌یافته تلقی می‌شود.



شکل (۳-۲۳): اتمام بارگذاری

۳-٤- بسته‌های TNS

اطلاعاتی که در این بخش ارائه می‌شود، نتیجه‌ی بررسی‌های نرم‌افزار wireshark و اطلاعات منتشرشده در مراجعی هستند که به صورت مهندسی معکوس استخراج شده‌اند.[۱]

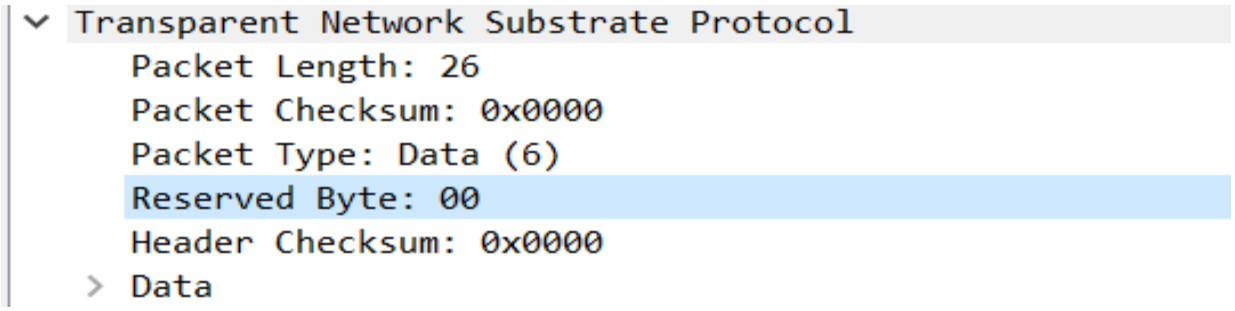
۳-٤-۱- سرآیند بسته‌ها

سرآیند بسته حافظه‌ی ثابتی دارد و اطلاعات آن مطابق جدول زیر نوشته شده است.

Total Header size: 8 Byte

Field	length	Summery
Length word	2 Byte (Word)	-
Checksum	2 Byte (Word)	-By default: 0x0000
Packet type	1 Byte (Byte)	-14 types
Flags (or Reserved Bytes)	1 Byte (Byte)	-usually unused
Header checksum	2 Byte (Word)	-By default: 0x0000 -usually unused

جدول (۳-۱): اطلاعات سرآیند بسته



شکل (۳-۲۲): سرآیند بسته‌ی داده در wireshark

۳-٤-۲- انواع بسته‌ها

در جدول زیر مهم‌ترین بسته‌های TNS قابل مشاهده است. مواردی که با علامت سوال مشخص شده است، در ابهام است.

Type	Packet name	Packet contain data?	Machin
1	Connect	Yes	client
2	Accept	Yes	server
3	Ack	No	client, server
4	Refuse	No	server
5	Redirect	Yes	server
6	Data	Yes	client, server
7	Null	No	?
9	Abort	?	?
11	Resend	No	server
12	Marker	Yes	client, server
13	Attention	?	?
14	Control	?	?

جدول (۳-۲): انواع بسته‌ها

۳-٤-۳- شرح چند بسته‌ی مهم

1-Connect:

در این بسته عمدتا اطلاعات مربوط به شروع ارتباط با مقصد که شامل موارد زیر است مشخص می‌شود:

- ورژن ارتباطی
- حداقل ورژن قابل قبول
- طول بسته‌ی connection
- حداکثر حجم داده رد و بدل شده
- نوع پروتوکل ارتباطی
- آدرس و پورت پایگاه داده‌ی مقصد
- محیط برنامه نویسی مبدا(در مثال SQL Developer)
- نام ماشین کاربر (در مثال DELL)
- نام دیتابیس سرویس‌دهنده (در مثال orcls)

نمونه بسته:

- **مواردی که پررنگ شده‌اند اطلاعات مهمی در بردارند.**

Packet Length: 249

Packet Checksum: 0x0000

Packet Type: Connect (1)

Reserved Byte: 00

Header Checksum: 0x0000

Connect

Version: 317

Version (Compatible): 300

Service Options: 0x0c41, Header Checksum, Full Duplex

Session Data Unit Size: 8192

Maximum Transmission Data Unit Size: 65535

NT Protocol Characteristics: 0x4f98, Confirmed release, Data test, Callback IO supported, ASync IO Supported, Packet oriented IO, Can grant connection to another, Generate SIGPIPE signal, Generate SIGURG signal

Line Turnaround Value: 0

Value of 1 in Hardware: 0001

Length of Connect Data: 179

Offset to Connect Data: 70

Maximum Receivable Connect Data: 0

Connect Flags 0: 0x81, NA services wanted

Connect Flags 1: 0x81, NA services wanted

Trace Cross Facility Item 1: 0x00000000

Trace Cross Facility Item 2: 0x00000000

Trace Unique Connection ID: 0x0000000000000000

Connect Data:
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=192.168.235.133)(PORT=1521))(CONNECT_DATA=(CID=(PROGRAM=SQL Developer)(HOST=__jdbc__)(USER=Dell))(SERVER=DEDICATED)(SERVICE_NAME=orcls)))

2-Accept:

نمونه بسته:

این بسته زمانی که listener درخواست اتصال را تایید می‌کند ارسال می‌شود.

Transparent Network Substrate Protocol

Packet Length: 32

Packet Checksum: 0x0000

Packet Type: Accept (2)

Reserved Byte: 00

Header Checksum: 0x0000

Accept

Version: 314

Service Options: 0x0841, Header Checksum

Session Data Unit Size: 8192

Maximum Transmission Data Unit Size: 65535

Value of 1 in Hardware: 0100

Accept Data Length: 0

Offset to Accept Data: 32

Connect Flags 0: 0xe1, NA services wanted

Connect Flags 1: 0x01, NA services wanted

3-Ack

قالبا برای تایید ارسال پیام‌ها استفاده می‌شود.

4- Refuse

چنان‌چه شنونده از سرویس یا بسته‌ای که مشتری به میزبان ارسال می‌کند، اطلاعی نداشته باشد، این بسته ارسال می‌شود.

5- Redirect

پس از ارسال بسته‌ی اتصال، چنان‌چه شنونده بخواهد مشتری را به درگاه یا پورت دیگری ارجاع بدهد، از این بسته استفاده می‌کند.

6- Data

این بسته پرکاربرد ترین بسته در پروتوکل محصوب می‌شود. تقریبا ۹۰ درصد بسته‌های ردوبدل شده بین مشتری و سرور، از این نوع می‌باشد. این بسته خود از چند نوع با شناسه مخصوص تشکیل می‌شود.

- بایت یازدهم از این بسته‌ها، مشخص می‌کند که بسته چه داده‌ای به همراه دارد:

- 0x01: از این بسته در مراحل مذاکره مشتری و سرویس‌دهنده استفاده می‌شود. به عنوان مثال، مشتری اطلاعات مربوط به ورژن پروتکل، تعداد مجموعه‌ی حروف استفاده شده و پرچم‌ها را برای سرویس دهنده ارائه می‌دهد.
- 0x02: اطلاعات مرتبط با نوع داده را نشان می‌دهد.
- 0x03: فراخوانی‌های توابع مهمی توسط این بسته‌ها انجام می‌شود. هر تابع کد مخصوصی دارد که در بایت بعدی مشخص می‌شود:

```
0×02      Open
0×03      Query
0×04      Execute
0×05      Fetch
0×08      Close
0×09      Disconnect/logoff
0×0C      Auto Commit ON
0×0D      Auto Commit OFF
0×0E      Commit
0×0F      Rollback
0×14      Cancel
0×2B      Describe
0×30      Startup
0×31      Shutdown
0×3B      Version
0×43      K2 Transactions
0×47      Query
0×4A      OSQL7
0×5C      OKOD
0×5E      Query
0×60      LOB Operations
0×62      ODNY
0×67      Transaction – end
0×68      Transaction – begin
0×69      OCCA
```

```
0×6D      Startup
0×51      Logon (present password)
0×52      Logon (present username)
0×73      Logon (present password - send AUTH_PASSWORD)
0×76      Logon (present username - request AUTH_SESSKEY)
0×77       Describe
0×7F      OOTCM
0×8B      OKPFC
```

شکل (۳-۲۳): کد اطلاعات گوناگون بسته‌ی داده براساس بایت ۱۲

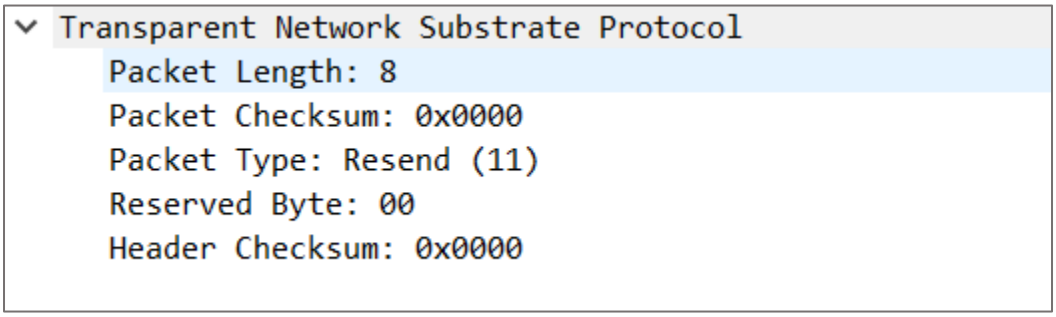
توضیحات بیشتر در مرجع موجود است.[۱]

7-Null

مشتریان گاهی برای حفظ ارتباط خود با سرور، این بسته را ارسال می‌کنند.

11-Resend

این بسته حاوی داده نمی‌باشد و تنها شامل یک سرآیند می‌شود که نوع بسته را مشخص می‌کند. این بسته از جانب سرور به مشتری ارسال می‌شود که درخواست (به طور معمول درخواست اتصال) خود را مجدد ارسال کند.

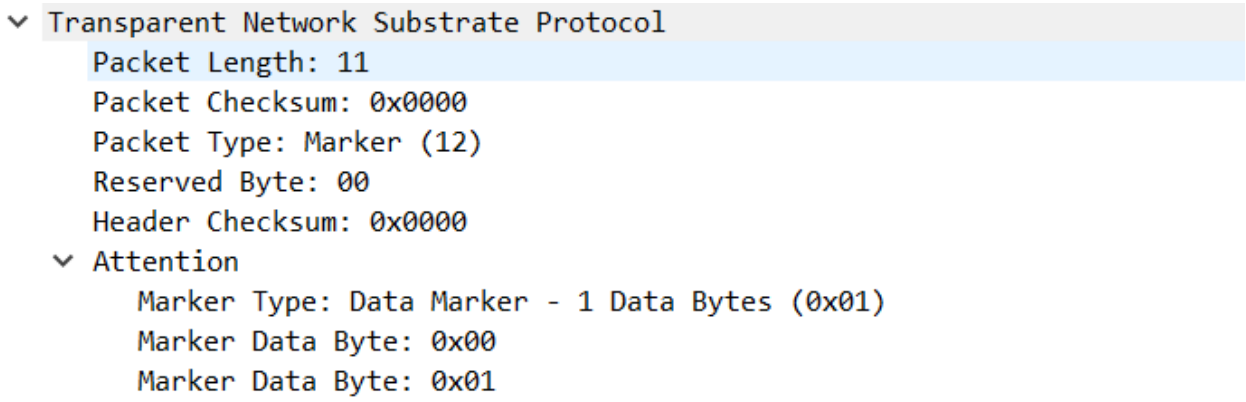


شکل (۳-۲٤): سرآیند بسته‌ی ارسال مجدد

12-Marker

پس از برقراری اتصال و شروع نشست، داده ها میان مشتری و سرور جابجا می‌شوند. این بسته معمولا در مرحله‌ی تبادل داده‌ها و ارسال کوئری‌ها از سمت سرور و مشتری انجام می‌شود. از این بسته به عنوان وقفه استفاده می‌شود و می‌تواند جریان ارسال داده را کنترل نماید.

نمونه بسته:



شکل (۳-۲۵): سرآیند و محتوای بسته‌ی Marker

به عنوان مثال مشتری پرسشی را به سمت سرور ارسال کرده است که در پایگاه داده، جدولی در رابطه با آن وجود ندارد. در پاسخ، سرور چند بسته ی بالا را ارسال می‌کند تا به دنبال آن جلوی ارسال داده از سوی مشتری را بگیرد و جست‌وجوی جدول را انجام دهد. سرانجام با مشخص شدن وضعیت درخواست، بسته‌ی "وضعیت" از سوی سرور ارسال می‌شود.

فصل چهارم

برنامه‌نویسی تعامل مشتری و سرویس‌دهنده

٤-١- مقدمه

در مواقعی لازم است تعامل میان مشتری و سرویس‌دهنده توسط کد یا سناریویی که برنامه‌نویس تعیین کرده است انجام شود. سپس با بررسی نرم‌افزار wireshark رفتار سیستم مورد بررسی قرار بگیرد. به عبارتی برنامه‌نویس سعی می‌کند سیستم را در عمل انجام شده قرار دهد. به عنوان مثال برنامه‌نویس برنامه‌ای بنویسد که یک مشتری یک بسته‌ی خالی و بلاتعریف را به پورت ۱۵۲۱ ارسال کند و پیغام های سرویس‌دهنده را ثبت کند. همزمان با نرم‌افزار wireshark بسته‌های جابه‌جا شده قابل بررسی هستند.

در این فصل سعی می‌شود یک پروکسی ساده جهت تعامل سرویس‌دهنده و مشتری پیاده‌سازی شود. گسترش این پروکسی می‌تواند به بهبود نتایج تحقیقات درباره پروتکل TNS کمک نماید.

٤-٢- زبان Golang و کتابخانه‌ی GoPacket

زبان گو زبانی مناسب برای برنامه‌نویسان شبکه است. به علت اینکه در محصول رایمون پیام‌پرداز از این زبان استفاده شده است، پس ترجیحا از این زبان برای ادامه آزمایش استفاده می‌شود. کتابخانه‌ی GoPacket یک کتابخانه‌ی مناسب برای کار با بسته‌ها است. از طریق این کتابخانه می‌توان بسته‌ی دلخواه ایجاد و ارسال کرد یا

بسته‌های جابه‌جا شده را بررسی یا دستکاری کرد. این کتابخانه به صورت پیشفرض همراه با زبان گو نصب می‌شود.

٤-٣- ایجاد یک پروکسی ساده

١- پیش از هرچیز باید بسته‌ی main و کتابخانه‌های مورد نیاز زیر را به پروژه اضافه کرد.

```
1   package main
2
3   import (
4       "bytes"
5       "fmt"
6       "log"
7       "time"
8
9       "github.com/google/gopacket"
10      "github.com/google/gopacket/pcap"
11  )
12
```

شکل (٤-١): کتابخانه‌ها

از ماژول Byte به منظور کار با داده‌های حافظه، fmt برای انجام عملیات رایج زبان گو (مانند دستور چاپ)، log برای ثبت خطاهای برنامه و time به منظور کار با زمان استفاده می‌شود. با استفاده از کتابخانه‌ی GoPacket می‌توان عملیات‌های متنوعی بر روی بسته‌ها انجام داد.

٢- در زبان گو می‌توان متغیرهای سراسری پروکسی را به صورت زیر تعریف نمود. این کار به جلوگیری از پیچیدگی کد کمک خواهد کرد.

```
13  var (
14      snapshot_len    int32 = 1024
15      promiscuous     bool  = false
16      err             error
17      timeout         time.Duration = 30 * time.Second
18      handle          *pcap.Handle
19      pac_counter     int32 = 0
20      character_indx  int   = 0
21      character       byte
22      username_string bytes.Buffer
23      query_string    bytes.Buffer
24      filter          string = "tcp and port 1521"
25  )
```

شکل (٤-٢): متغیرها

از متغیرهای سراسری مهم می‌توان به بافر نام‌کاربری، بافر پرسش، زمان اتمام و ارور اشاره کرد.

۳- ساختار کلی برنامه داخل یک تابع main نوشته می‌شود که طی یک حلقه‌ی While جریان پیدا می‌کند.

```
27  func main() {
28      devices := Scan_AllDevices()
29      // Open device
30      handle := Open_Device(devices[0].Name)
31
32      defer handle.Close()
33      // Set filter
34      handle.SetBPFFilter(filter)
35      // Use the handle as a packet source to process all packets
36      fmt.Println("Snifing is starting")
37      packetSource := gopacket.NewPacketSource(handle, handle.LinkType())
38      for packet := range packetSource.Packets() {
39          // Process packet here
40          character_indx = 0
41          Check_username_pack(packet)
42          Check_OCI_pack(packet)
43
44      }
45
46  }
```

شکل (٤-٣): تابع main

- ابتدا با استفاده از تابع Scan_AllDevices() تمامی ماشین‌های شبکه در قالب یک لیست شناسایی می‌شود.
- ماشین سرویس‌دهنده، اولین ماشین در این لیست است. با استفاده از تابع Open_Device(string) نام سرویس‌دهنده به تابع داده می‌شود و سرویس‌دهنده باز می‌شود. خروجی این تابع یک handler است.
- فیلتر (tcp and port 1521) بر روی hanler اعمال می‌شود تا بسته‌هایی که به پورت ۱۵۲۱ سرویس‌دهنده رفت‌وآمد می‌کنند بررسی شود.
- این handler به عنوان یک مرجع بسته، به یک PacketSource واگذار می‌شود.
- یک حلقه for، تمامی بسته‌هایی که از سمت مرجع ایجاد می‌شود را ردیابی می‌کند. داخل حلقه for می‌توان کدهای و عملیات دلخواه بر روی بسته‌ها را نوشت.
- بسته در مرحله اول بررسی می‌شود که آیا حاوی نام‌کاربری هست یا خیر. در صورت وجود نام کاربری را ثبت می‌کند.
- در مرحله دوم بررسی می‌شود که آیا بسته حاوی پرسش‌کاربر (User OCI) هست یا خبر. در صورت وجود، پرسش را ثبت می‌کند.

٤- در ادامه نوشتن توابع زیر مورد نیاز است. تابع Scan_AllDevices() تمامی ماشین‌های فعال را شناسایی می‌کند. در صورت بروز خطا، پیغام خطا ثبت می‌شود. خروجی تابع، لیستی از دستگاه‌ها است.

```
48  func Scan_AllDevices() []pcap.Interface {
49      devices, err := pcap.FindAllDevs()
50      if err != nil {
51          log.Fatal(err)
52      }
53      return devices
54
55  }
```

شکل (٤-٤): Scan_AllDevices

۵- تابع Open_Device()، نام دستگاه را به عنوان رشته‌ی ورودی دریافت می‌کند و آن را باز می‌کند و یک کنترلگر به عنوان خروجی برمی‌گرداند.

```
57  func Open_Device(Dvice_name string) *pcap.Handle {
58      handle, err = pcap.OpenLive(Dvice_name, snapshot_len, promiscuous, timeout)
59      if err != nil {
60          log.Fatal(err)
61      }
62      return handle
63  }
```

شکل (٤-٥): Open_Device

٦- طبق بررسی‌ها بسته‌هایی که شناسایی می‌شوند، در حالت عادی ۳ لایه دارند. در صورتی که لایه‌ی چهارم شناسایی شود، این بسته حتما TNS است.

```
65  func Packet_isTNS(packet gopacket.Packet) (result bool) {
66      if len(packet.Layers()) == 4 {
67          return true
68      }
69      return false
70  }
```

شکل (٤-٦): Packet_isTNS

۷- در این تابع، وجود نام‌کاربری در یک بسته‌ی TNS مشخص می‌شود. ابتدا این شرط بررسی می‌شود که اندازه‌ی حافظه‌ی بسته‌ی TNS از ۱۰ بایت بیشتر باشد. پس از آن، طبق بررسی بایت ۱۱ و ۱۲ بر اساس شکل (۳-۲۳) می‌توان تشخیص داد که این بسته حاوی نام کاربری است یا خیر. در صورت فراهم بودن شرایط، مقدار true باز گردانده می‌شود.

```
72  func Username_exist(packet gopacket.Packet) (result bool) {
73      if len(packet.Layers()[3].LayerContents()) > 10 &&
74          packet.Layers()[3].LayerContents()[10] == 0x03 &&
75          packet.Layers()[3].LayerContents()[11] == 0x76 {
76          return true
77      }
78      return false
79  }
```

شکل (٤-٧): Username_exist

۸- در ادامه تابعی برای تشخیص اعداد اسکی استاندارد نگارشی، نوشته می‌شود. این تابع یک بایت را دریافت می‌کند و کاراکتر بودن یا نبودن آن را ارزیابی می‌کند.

```
81  func Standard_character(character byte) (result bool) {
82      if character > 31 && character < 127 {
83          return true
84      }
85      return false
86  }
```

شکل (٤-٨): Standard_character

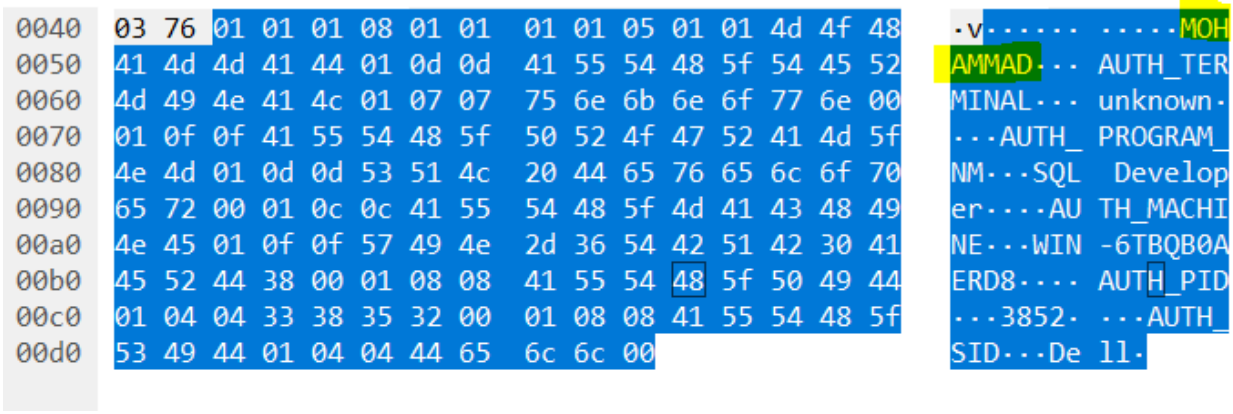
۹- مشابه کاری که در تابع قبل انجام شد، ابتدا بسته از لحاظ اندازه‌ی حافظه تایید می‌شود. سپس شروط وجود پرسش OSI کاربر بررسی می‌شود. در صورت وجود شرایط، مقدار true بازگردانده می‌شود.

```
88  func OCIQuery_exist(packet gopacket.Packet) (result bool) {
89      if len(packet.Layers()[3].LayerContents()) > 12 &&
90       ((packet.Layers()[3].LayerContents()[10] == 0x03 && packet.Layers()[3].LayerContents()[11] == 0x5e
91       (packet.Layers()[3].LayerContents()[10] == 0x11 && packet.Layers()[3].LayerContents()[11] == 0x69)
92          return true
93      }
94      return false
95  }
```

شکل (٤-٩): OCIQuery_exist

۱۰- این تابع یک بسته را دریافت می‌کند و در صورت وجود شرایط لازم برای داشتن username، اطلاعات آن را نمایش می‌دهد. ابتدا تعیین می‌شود که آیا بسته مورد نظر، مربوط به پروتکل TNS است یا خیر. پس از آن شرط داشتن نام‌کاربری بررسی می‌شود.

پس از آن به کمک یک متغیر، بایت‌های غیرمتعارف در حلقه‌ی اول رد می‌شود و پس از رسیدن به اولین کاراکتر استاندارد، نام کاربری یافت می‌شود.



شکل (٤-١٠): نام کاربری

```go
func Log_username_pack(packet gopacket.Packet) {
    if Packet_isTNS(packet) && Username_exist(packet) {
        pac_counter += 1
        fmt.Println("Packet with username founded :", pac_counter)
        fmt.Println(packet.Layers()[3].LayerContents())
        character_indx = 12
        character = packet.Layers()[3].LayerContents()[character_indx]
        for Standard_character(character) == false || string(character) == "!" {
            character_indx++
            character = packet.Layers()[3].LayerContents()[character_indx]
        }
        for Standard_character(character) {
            username_string.WriteByte(character)
            character_indx++
            character = packet.Layers()[3].LayerContents()[character_indx]
        }
        fmt.Println("username founded :", username_string.String())
        fmt.Println("########################################")
        username_string.Reset()
    }

}
```

شکل (٤-١١): Log_username_pack

١١- برای نمایش بسته‌های OCI، مطابق تابع قبل برنامه نوشته می‌شود. با این تفاوت که ازآنجایی که درخواست OCI در مکان مشخصی قرار ندارد، پس به ناچار کل محتویات بسته به عنوان خروجی چاپ می‌شود.

```
120  func Log_OCI_pack(packet gopacket.Packet) {
121      if Packet_isTNS(packet) && OCIQuery_exist(packet) {
122          pac_counter += 1
123          fmt.Println("Packet with query founded :", pac_counter)
124          fmt.Println(packet.Layers()[3].LayerContents())
125          fmt.Println("########################################")
126          character_indx = 12
127          character = packet.Layers()[3].LayerContents()[character_indx]
128          for character_indx < len(packet.Layers()[3].LayerContents())-1 {
129              query_string.WriteByte(character)
130              character_indx++
131              character = packet.Layers()[3].LayerContents()[character_indx]
132          }
133          fmt.Println("Query founded :", query_string.String())
134          query_string.Reset()
135
136      }
137
138  }
```

شکل (٤-١٢): Log_OCI_pack

٤-٤- اجرای برنامه

برای اجرای برنامه باید یک ترمینال باز کرده و با دستور CD وارد پوشه‌ی برنامه شد. سپس با دستور زیر برنامه اجرا می‌شود:

go run program_name.go

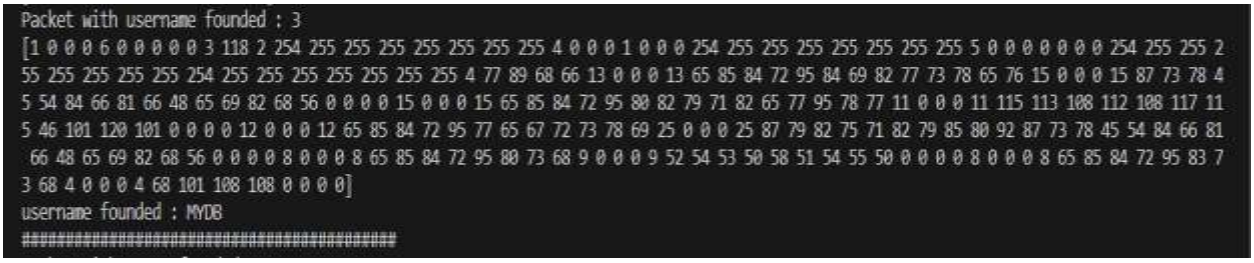
```
PS C:\Users\Dell\Desktop\go_proj> go run test.go
Snifing is starting
```

شکل (٤-١٣): شروع اجرا

اکنون زمان آن است که کاربر وارد سیستم شود. چنانچه کاربر با سمت کاربر اصلی یا عادی وارد شود، نام کاربری او نمایش داده می‌شود.

```
Packet with username founded : 1
[0 255 0 0 6 0 0 0 0 0 3 118 2 254 255 255 255 255 255 255 255 3 0 0 0 33 0 0 0 254 255 255 255 255 255 255 255 5 0 0 0 0 0 0 0
254 255 255 255 255 255 255 255 254 255 255 255 255 255 255 255 3 83 89 83 13 0 0 0 13 65 85 84 72 95 84 69 82 77 73 78 65 76 15
 0 0 0 15 87 73 78 45 54 84 66 81 66 48 65 69 82 68 56 0 0 0 0 15 0 0 0 15 65 85 84 72 95 80 82 79 71 82 65 77 95 78 77 11 0 0 0
 11 115 113 108 112 108 117 115 46 101 120 101 0 0 0 0 12 0 0 0 12 65 85 84 72 95 77 65 67 72 73 78 69 25 0 0 0 25 87 79 82 75 7
1 82 79 85 80 92 87 73 78 45 54 84 66 81 66 48 65 69 82 68 56 0 0 0 0 8 0 0 0 8 65 85 84 72 95 80 73 68 9 0 0 0 9 53 53 54 52 58
 53 50 57 50 0 0 0 0 8 0 0 0 8 65 85 84 72 95 83 73 68 4 0 0 0 4 68 101 108 108 0 0 0 0]
username founded : SYS
########################################
```

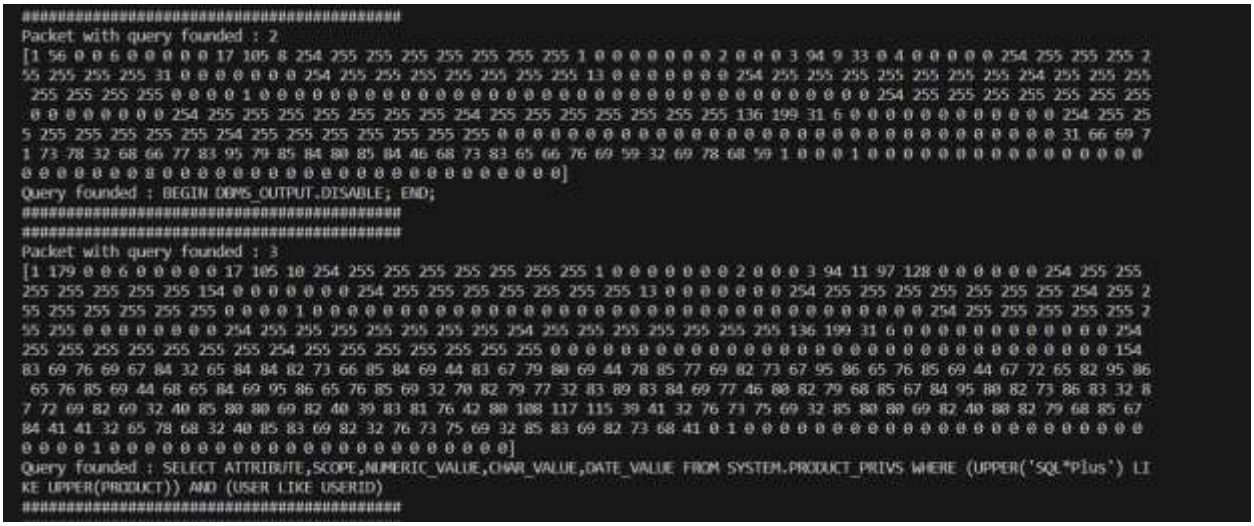
شکل (٤-١٤): نام‌کاربری مدیر



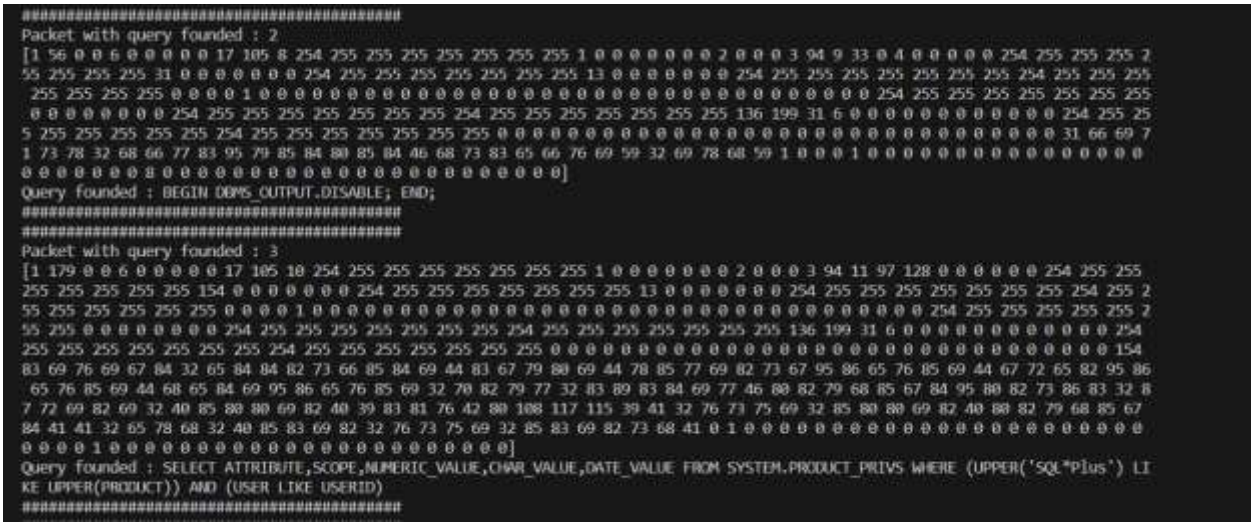
شکل (٤-١٥): نام‌کاربری عادی

چنانچه پرسش زیر در کنسول یا محیط SQLdeveloper نوشته شود، بسته‌ی مربوطه به این شکل ثبت می‌شود.

select vid_id from video where vid_id = 54



شکل (٤-١٦): پرسش کاربر



شکل (٤-١٧): پرسش کاربر

فصل پنجم

جمع‌بندی

۵-۱-نتیجه‌گیری

همانطور که در فصل اول اشاره شد، پروتکل TNS یک پروتکل اختصاصی و بسته برای سیستم‌های مبتنی بر پایگاه داده‌ی اوراکل می‌باشد. پروتکل علیرغم نقاط قوت بسیاری که دارد، در مواردی با چالش‌ها و حفره‌هایی مواجه است که درنظر گرفتن آن برای متخصصین امنیت حائز اهمیت است.

متاسفانه مراجع و منابع موجود از قاطعیت چندانی برخوردار نیستند و این موضوع یک چالش برای افرادیست که به دنبال تحقیق و توسعه‌ی نرم‌افزارهای حوزه‌ی امنیت هستند. محصول رایمون شرکت پیام‌پرداز از آنجایی که یک نرم‌افزار از خانواده‌ی Pam است، برای پشتیبانی از پروتکل‌های گوناگون نیازمند اطلاعات کافی از جایگاه و عملکرد آنان می‌باشد.

با این اوصاف محدودیت‌ها به معنای توقف پژوهش نمی‌باشد. در اکثر مواقع، بسیاری از شرکت‌ها به دلایل مختلفی چون حفظ امنیت محصولاتشان، شرایط رقابتی بازار یا جلوگیری از هرگونه اقدام خرابکارانه جلوی نشر اطلاعات را می‌گیرند. پس تنها راهی که باقی می‌ماند در گام اول، جمع آوری تجربیات و اطلاعات موجود و سپس بررسی صحت آن‌ها از طریق روش‌های آزمایشگاهی است. با این وجود ممکن است بسیاری از سوالات حتی به کمک روش‌های مهندسی معکوس یافت نشود. چه بسا اطلاعاتی که گذشتگان به آن دست یافته‌اند،

حاصل همین آزمایشات و حدس و گمان‌ها است. اما چیزی که حائز اهمیت می‌باشد آن است که رشته‌ی تحقیقات نباید متوقف شود و ادامه‌ی آن می‌بایست توسط دیگران انجام شود.

۵-۲-نقاط ابهام و موارد قابل پیشرفت

۱- در فصل اول گزارش به این مسئله اشاره شد که از الگوریتم تغییر یافته‌ی DES برای رمزنگاری بعضی از داده‌های حیاتی مانند رمز عبور استفاده می‌شود. با این اوصاف به علت مشخص نبودن کلید الگوریتم به منظور رمزگشایی آن، هنوز این مسئله در ابهام است.

۲- با توجه به تحقیقات انجام شده، اطلاعات مهمی مانند نام‌سرویس‌دهنده، کلید نشست و نام پایگاه داده مورد درخواست از طریق نرم‌افزار wireshark قابل استخراج است. با این حال درباره‌ی نحوه‌ی استخراج و بازیابی رمزعبور کاربر تاکنون نتیجه‌ای حاصل نشده است

۳- در فصل سوم اشاره شد که حدود ۹۰ درصد بسته‌های جابه‌جا شده در پروتکل TNS، مبتنی بر بسته‌های نوع ٦ (Data) است. در برخی مراجع به بسته‌های دیگری مانند Attention، Abrot یا Control اشاره شده است که در سناریوهای مختلف Wireshark یافت نشده‌اند. همچنین مواردی که در بخش انواع بسته‌ها به عنوان علامت سوال مطرح شده است، مورد ابهام می‌باشد.

٤- از مواردی که می‌توان به پروکسی اضافه نمود، یافتن بسته‌ی حاوی رمزعبور کاربر و رمزگشایی آن می‌باشد. درباره‌ی موضوع ثبت خروجی پرسش، برخی از اطلاعات پاسخ سرور مانند اعداد رمز می‌شود یا قابل استخراج از بسته‌های OPI نیستند. با این حال پاسخ‌هایی که با فرمت رشته نوشته شده باشند، در بسته‌های OPI قابل شناسایی هستند.

منابع

[1] - The oracle hackers handbook, David Litchfield

[2] - Database Net Services Administrator's Guide- https://docs.oracle.com/en/database/oracle/oracle-database/19/netag/database-net-services-administrators-guide.pdf

[3] - https://wirexsystems.com/resource/protocols/tns

[4] - https://www.relationaldbdesign.com/network-topology/module3/transparent-network-substrate.php

[5] - https://flylib.com/books/en/2.680.1/the_oracle_network_architecture.html