本章 LyShark 将带大家学习如何在内核中使用标准的 Socket 套接字通信接口，我们都知道 Windows 应用层下可直接调用 WinSocket 来实现网络通信，但在内核模式下应用层API接口无法使用，内核模式下有一套专有的 WSK 通信接口，我们对WSK进行封装，让其与应用层调用规范保持一致，并实现内核与内核直接通过 Socket 通信的案例。

# 当然在早期如果需要实现网络通信一般都会采用 TDI 框架，但在新版本 Windows10 系统上虽然依然可以使用TDI接口，但是 LyShark 并不推荐使用，因为微软已经对接口搁置了，为了使WSK通信更加易用， 我们需要封装内核层中的通信API，新建 LySocket.hpp 头文件，该文件中封装了WSK通信API接口，其封装格式与应用层接口保持了高度一致，当需要在内核中使用Socket通信时可直接引入本文件。

我们需要使用 WDM 驱动程序，并配置以下参数。

配置属性 -> 连接器 -> 输入-> 附加依赖 -> $(DDK\_LIB\_PATH)\Netio.lib

配置属性 -> C/C++ -> 常规 -> 设置 警告等级2级 (警告视为错误关闭)

配置好以后，我们就开始吧，先来看看服务端如何实现！

对于 服务端 来说，驱动通信必须保证服务端开启多线程来处理异步请求，不然驱动加载后系统会处于等待状态，而一旦等待则系统将会卡死，那么对于服务端 DriverEntry 入口说我们不能让其等待，必须使用 PsCreateSystemThread 来启用系统线程，该函数属于WDM的一部分，官方定义如下；

NTSTATUS PsCreateSystemThread(

[out]

[in]

[in, optional] [in, optional]

PHANDLE

ULONG

ThreadHandle,

DesiredAccess,

POBJECT\_ATTRIBUTES ObjectAttributes,

HANDLE

[out, optional] PCLIENT\_ID [in] PKSTART\_ROUTINE

[in, optional] PVOID

ProcessHandle, ClientId, StartRoutine,

StartContext

);

我们使用 PsCreateSystemThread 函数开辟线程 TcpListenWorker 在线程内部执行如下流程启动驱动服务端，由于我们自己封装实现了标准接口组，所以使用起来几乎与应用层无任何差异了。

# CreateSocket 创建套接字

Bind 绑定套接字

Accept 等待接收请求

Receive 用于接收返回值

Send 用于发送返回值

// 署名权

// right to sign one's name on a piece of work

// PowerBy: LyShark

// Email: [me@lyshark.com](mailto:me@lyshark.com)

#include "LySocket.hpp"

PETHREAD m\_EThread = NULL;

// 线程函数

// PowerBy: LySHark

VOID TcpListenWorker(PVOID Context)

{

WSK\_SOCKET paccept\_socket = NULL; SOCKADDR\_IN LocalAddress = { 0 }; SOCKADDR\_IN RemoteAddress = { 0 }; NTSTATUS status = STATUS\_UNSUCCESSFUL;



// 创建套接字

PWSK\_SOCKET TcpSocket = CreateSocket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP, WSK\_FLAG\_LISTEN\_SOCKET);

if (TcpSocket == NULL)

{

return;

}

// 设置绑定地址

LocalAddress.sin\_family = AF\_INET; LocalAddress.sin\_addr.s\_addr = INADDR\_ANY; LocalAddress.sin\_port = HTON\_SHORT(8888);

status = Bind(TcpSocket, (PSOCKADDR)&LocalAddress); if (!NT\_SUCCESS(status))

{

return;

}

// 循环接收

while (1)

{

CHAR read\_buffer = (CHAR )ExAllocatePoolWithTag(NonPagedPool, 2048, "read");

paccept\_socket = Accept(TcpSocket, (PSOCKADDR)&LocalAddress, (PSOCKADDR)&RemoteAddress);

if (paccept\_socket == NULL)

{

continue;

}

// 接收数据

memset(read\_buffer, 0, 2048);

int read\_len = Receive(paccept\_socket, read\_buffer, 2048, 0); if (read\_len != 0)

{

DbgPrint("[内核A] => %s \n", read\_buffer);

// 发送数据

char send\_buffer[2048] = "Hi, lyshark.com B"; Send(paccept\_socket, send\_buffer, strlen(send\_buffer), 0);

// 接收确认包

memset(read\_buffer, 0, 2048);

Receive(paccept\_socket, read\_buffer, 2, 0);

}

// 清理堆

if (read\_buffer != NULL)

{

ExFreePool(read\_buffer);

}



// 关闭当前套接字

if (paccept\_socket)

{

CloseSocket(paccept\_socket);

}

}

if (TcpSocket)

{

CloseSocket(TcpSocket);

}

PsTerminateSystemThread(STATUS\_SUCCESS); return;

}

// 关闭套接字

VOID UnDriver(PDRIVER\_OBJECT driver)

{

WSKCleanup();

KeWaitForSingleObject(m\_EThread, Executive, KernelMode, FALSE, NULL); if (m\_EThread != NULL)

{

ObDereferenceObject(m\_EThread);

}

}

NTSTATUS DriverEntry(IN PDRIVER\_OBJECT Driver, PUNICODE\_STRING RegistryPath)

{

DbgPrint("hello lyshark.com \n");

// 初始化

WSKStartup();

HANDLE hThread = NULL;

NTSTATUS status = STATUS\_UNSUCCESSFUL;

// 创建系统线程

status = PsCreateSystemThread(&hThread, THREAD\_ALL\_ACCESS, NULL, NULL, NULL, TcpListenWorker, NULL);

if (!NT\_SUCCESS(status))

{

return status;

}

// 获取线程EProcess结构

status = ObReferenceObjectByHandle(hThread, THREAD\_ALL\_ACCESS, NULL, KernelMode, (PVOID )&m\_EThread, NULL);

if (NT\_SUCCESS(status) == FALSE)

{

return status;

}

ZwClose(hThread);

Driver->DriverUnload = UnDriver;

return STATUS\_SUCCESS;

}

# 对于客户端来说，只需要创建套接字并连接到指定地址即可，这个过程大体上可以总结为如下；

CreateSocket 创建套接字Bind 绑定套接字Connect 链接服务端驱动

Send 发送数据到服务端

Receive 接收数据到服务端

// 署名权

// right to sign one's name on a piece of work

// PowerBy: LyShark

// Email: [me@lyshark.com](mailto:me@lyshark.com)

#include "LySocket.hpp"

VOID UnDriver(PDRIVER\_OBJECT driver)

{

// 卸载并关闭Socket库

WSKCleanup();

}

NTSTATUS DriverEntry(IN PDRIVER\_OBJECT Driver, PUNICODE\_STRING RegistryPath)

{

DbgPrint("hello lyshark.com \n");

// 初始化

WSKStartup();

NTSTATUS status = STATUS\_SUCCESS;

SOCKADDR\_IN LocalAddress = { 0, }; SOCKADDR\_IN RemoteAddress = { 0, };

// 创建套接字

PWSK\_SOCKET TcpSocket = CreateSocket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP, WSK\_FLAG\_CONNECTION\_SOCKET);

if (TcpSocket == NULL)

{

Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

LocalAddress.sin\_family = AF\_INET; LocalAddress.sin\_addr.s\_addr = INADDR\_ANY;

status = Bind(TcpSocket, (PSOCKADDR)&LocalAddress);

// 绑定失败则关闭驱动

if (!NT\_SUCCESS(status))

{

CloseSocket(TcpSocket);

Driver->DriverUnload = UnDriver;

return STATUS\_SUCCESS;



}

// 初始化服务端地址与端口信息

ULONG address[4] = { 127, 0, 0, 1 };

RemoteAddress.sin\_family = AF\_INET;

RemoteAddress.sin\_addr.s\_addr = change\_uint(address[0], address[1], address[2], address[3]);

RemoteAddress.sin\_port = HTON\_SHORT(8888);

status = Connect(TcpSocket, (PSOCKADDR)&RemoteAddress);

// 连接服务端,如果失败则关闭驱动

if (!NT\_SUCCESS(status))

{

CloseSocket(TcpSocket);

Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

// 发送数据

char send\_buffer[2048] = "hello lyshark.com A"; Send(TcpSocket, send\_buffer, strlen(send\_buffer), 0);

// 接收数据

CHAR read\_buffer = (CHAR )ExAllocatePoolWithTag(NonPagedPool, 2048, "read");

memset(read\_buffer, 0, 1024);

Receive(TcpSocket, read\_buffer, 2048, 0);

DbgPrint("[内核B] => %s \n", read\_buffer);

// 发送确认包

Send(TcpSocket, "ok", 2, 0);

// 释放内存ExFreePool(read\_buffer); CloseSocket(TcpSocket);

Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

编译两个驱动程序，首先运行 server.sys 驱动，运行后该驱动会在后台等待客户端连接，接着运行

client.sys 屏幕上可输出如下提示，说明通信已经建立了。

