

Forum:[Hack Forums](#)

Thread: [Low Level Win 11 socket POC // AFD.sys method // No Winsock](#)

Hackers - October 21, 2025, 8:25 pm

Low Level AFD.sys
Socket Creation
for Windows 11

I found this write-up about some undocumented functions in Windows.
This method uses AFD.sys on Win11, and does not use Winsock to open a socket.

Further research with IOCTL can be done to identify how the socket can be used.

This guide can help blue teams & cybersecurity researchers defend their machines by teaching them more about the lesser known & undocumented AFD.sys interface!

Software developers can write more efficient code, low level, and without relying on a boring library like Winsock. I'll be interested in hearing what people can discover or learn from this.

Read the Paper:

<https://github.com/LNodesL/files/blob/main/random/2025-07-16%20-%20Under%20the%20Hood%20of%20AFD.sys%20-%20Investigating%20Undocumented%20Interfaces.pdf>

Mirror:

<https://mega.nz/file/AvUESQLS#6dUV8TN-cOfG9wjRPm-oCPB9nA9JY33ahyw0PdI36ak>

Similar articles:

<https://leftarcode.com/posts/afd-reverse-engineering-part1/>

<https://leftarcode.com/posts/afd-reverse-engineering-part2/>

My Golang example:

<https://github.com/LNodesL/Golang-AFD.sys-Socket-Creation>

Here is the example code below that they gave. Keep in mind though that your machine's values may be different, and you shouldn't run this without reading the attached PDF first...

Spoiler ([Click to View](#))

```
Code [h1]include <stdint.h>[/h1]
[h1]include <Windows.h>[/h1]
[h1]include <winternl.h>[/h1]
[h1]include <iostream>[/h1]
[h1]pragma comment(lib, "ntdll.lib")[/h1]
enum AFD_ENDPOINT_FLAGS : uint32_t {
```

```

AFD_ENDPOINT_FLAG_CONNECTIONLESS = 0x0000000000000001,
AFD_ENDPOINT_FLAG_MESSAGE_MODE = 0x0000000000000010,
AFD_ENDPOINT_FLAG_RAW = 0x00000000000001000,
AFD_ENDPOINT_FLAG_MULTIPOINT = 0x00000000000010000,
AFD_ENDPOINT_FLAG_CROOT = 0x000000010000000,
AFD_ENDPOINT_FLAG_DROOT = 0x000000100000000,
AFD_ENDPOINT_FLAG_IGNORETDI = 0x0010000000000,
AFD_ENDPOINT_FLAG_RIOSOCKET = 0x0100000000000,
};
struct AFD_OPEN_PACKET_EA {
    uint32_t nextEntryOffset;
    uint8_t flags;
    uint8_t eaNameLength;
    uint16_t eaValueLength;
    char eaName[0x10];
    uint32_t endpointFlags;
    uint32_t groupID;
    uint32_t addressFamily;
    uint32_t socketType;
    uint32_t protocol;
    uint32_t sizeOfTransportName;
    uint8_t unknownBytes[0x9];
};
NTSTATUS createAfdSocket(PHANDLE socket) {
    const char* eaName = "AfdOpenPacketXX";
    UNICODE_STRING devName;
    RtlInitUnicodeString(&devName, L"\\Device\\Afd\\Endpoint");

    OBJECT_ATTRIBUTES object;
    object = { 0 };
    object.ObjectName = &devName;
    object.Length = 48;
    object.Attributes = 0x40;
    AFD_OPEN_PACKET_EA afdOpenPacketEA;
    afdOpenPacketEA.nextEntryOffset = 0x00;
    afdOpenPacketEA.flags = 0x00;
    afdOpenPacketEA.eaNameLength = 0x0F;
    afdOpenPacketEA.eaValueLength = 0x1e;
    afdOpenPacketEA.endpointFlags = 0x00;
    afdOpenPacketEA.groupID = 0x00;
    afdOpenPacketEA.addressFamily = AF_INET;
    afdOpenPacketEA.socketType = SOCK_STREAM;
    afdOpenPacketEA.protocol = IPPROTO_TCP;
    afdOpenPacketEA.sizeOfTransportName = 0x00;
    memset(afdOpenPacketEA.eaName, 0x00, 0x10);
    memcpy(afdOpenPacketEA.eaName, eaName, 0x10);
    memset(afdOpenPacketEA.unknownBytes, 0xFF, 0x9);
    IO_STATUS_BLOCK IoStatusBlock;
    return NtCreateFile(socket, GENERIC_READ | GENERIC_WRITE | SYNCHRONIZE, &object,
        &IoStatusBlock, 0, 0, FILE_SHARE_READ | FILE_SHARE_WRITE,
        FILE_OPEN_IF,
        FILE_SYNCHRONOUS_IO_NONALERT, &afdOpenPacketEA,
        sizeof(afdOpenPacketEA));

```

```
}
int main() {
    HANDLE socket;
    NTSTATUS status = createAfdSocket(&socket);
    if (!NT_SUCCESS(status)) {
        std::cout << "[-] Could not create socket: " << std::hex << status <<
std::endl;
        return 1;
    }
    std::cout << "[+] Socket created!" << std::endl;
    return 0;
}
Read and learn,
Be a legend.
```

PhillyStyle - October 21, 2025, 8:45 pm

Oh this is very interesting! PDF Downloaded for sure.

Hackers - October 21, 2025, 9:16 pm

(Oct 21, 2025 08:45 PM)PhillyStyle Wrote: [\[CITE\]](#) Oh this is very interesting! PDF Downloaded for sure.

:D happy researching!

You will need to do a good amount of work for real connections, but it's possible.

Please be very careful... low level work can be messy if you don't undo certain changes, or if you mess with your machine too much!

Phantom_NiNja - October 23, 2025, 3:42 pm

(Oct 21, 2025 08:25 PM)enki Wrote: [\[CITE\]](#) Low Level AFD.sys
Socket Creation
for Windows 11

I found this write-up about some undocumented functions in Windows.

This method uses AFD.sys on Win11, and does not use Winsock to open a socket.

Further research with IOCTL can be done to identify how the socket can be used.

This guide can help blue teams & cybersecurity researchers defend their machines by teaching them more about the lesser known & undocumented AFD.sys interface!

Software developers can write more efficient code, low level, and without relying on a boring library like Winsock. I'll be interested in hearing what people can discover or learn from this.

Read the Paper:

<https://github.com/LNodesL/blue-team-defense-research/blob/f043f9b5ba79aa8eaa1cce75b7ca0cc761964c12/Windows/2025-07-16%20-%20Under%20the%20Hood%20of%20AFD.sys%20-%20Investigating%20Undocumented%20Interfaces.pdf>

Mirror:

<https://mega.nz/file/AvUESQLS#6dUV8TN-cOfG9wjRPm-oCPB9nA9JY33ahyw0PdI36ak>

Here is the example code below that they gave. Keep in mind though that your machine's values may be different, and you shouldn't run this without reading the attached PDF first...

Spoiler ([Click to View](#))

```
Code [h1]include <stdint.h>[/h1]
[h1]include <Windows.h>[/h1]
[h1]include <winternl.h>[/h1]
[h1]include <iostream>[/h1]
[h1]pragma comment(lib, "ntdll.lib")[/h1]
enum AFD_ENDPOINT_FLAGS : uint32_t {
    AFD_ENDPOINT_FLAG_CONNECTIONLESS = 0x0000000000000001,
    AFD_ENDPOINT_FLAG_MESSAGE_MODE = 0x0000000000000010,
    AFD_ENDPOINT_FLAG_RAW = 0x0000000000000100,
    AFD_ENDPOINT_FLAG_MULTIPOINT = 0x0000000000001000,
    AFD_ENDPOINT_FLAG_CROOT = 0x0000000100000000,
    AFD_ENDPOINT_FLAG_DROOT = 0x0000010000000000,
    AFD_ENDPOINT_FLAG_IGNORETDI = 0x0010000000000000,
    AFD_ENDPOINT_FLAG_RIOSOCKET = 0x0100000000000000,
};
struct AFD_OPEN_PACKET_EA {
    uint32_t nextEntryOffset;
    uint8_t flags;
    uint8_t eaNameLength;
    uint16_t eaValueLength;
    char eaName[0x10];
    uint32_t endpointFlags;
    uint32_t groupID;
    uint32_t addressFamily;
    uint32_t socketType;
    uint32_t protocol;
    uint32_t sizeOfTransportName;
    uint8_t unknownBytes[0x9];
};
NTSTATUS createAfdSocket(PHANDLE socket) {
    const char* eaName = "AfdOpenPacketXX";
    UNICODE_STRING devName;
    RtlInitUnicodeString(&devName, L"\\Device\\Afd\\Endpoint");

    OBJECT_ATTRIBUTES object;
    object = { 0 };
    object.ObjectName = &devName;
    object.Length = 48;
    object.Attributes = 0x40;
    AFD_OPEN_PACKET_EA afdOpenPacketEA;
    afdOpenPacketEA.nextEntryOffset = 0x00;
    afdOpenPacketEA.flags = 0x00;
    afdOpenPacketEA.eaNameLength = 0x0F;
    afdOpenPacketEA.eaValueLength = 0x1e;
```

```

afdOpenPacketEA.endpointFlags = 0x00;
afdOpenPacketEA.groupID = 0x00;
afdOpenPacketEA.addressFamily = AF_INET;
afdOpenPacketEA.socketType = SOCK_STREAM;
afdOpenPacketEA.protocol = IPPROTO_TCP;
afdOpenPacketEA.sizeOfTransportName = 0x00;
memset(afdOpenPacketEA.eaName, 0x00, 0x10);
memcpy(afdOpenPacketEA.eaName, eaName, 0x10);
memset(afdOpenPacketEA.unknownBytes, 0xFF, 0x9);
IO_STATUS_BLOCK IoStatusBlock;
return NtCreateFile(socket, GENERIC_READ | GENERIC_WRITE | SYNCHRONIZE, &object,
&IoStatusBlock, 0, 0, FILE_SHARE_READ | FILE_SHARE_WRITE,
FILE_OPEN_IF,
FILE_SYNCHRONOUS_IO_NONALERT, &afdOpenPacketEA,
sizeof(afdOpenPacketEA));
}
int main() {
HANDLE socket;
NTSTATUS status = createAfdSocket(&socket);
if (!NT_SUCCESS(status)) {
std::cout << "[-] Could not create socket: " << std::hex << status <<
std::endl;
return 1;
}
std::cout << "[+] Socket created!" << std::endl;
return 0;
}
Read and learn,
Be a legend.

```

Thanks for the share. I book marked it. I will look into it later but very interesting.

I learned ASM 86-64bit and right now want to make stuff at the lower level.
I made my own apps but using the API. I did just just to learn what is commonly done.
I am right now slowly trying to learn how to not use the API and try to do my own thing directly with the kernel.

I seen this file being used with process monitoring app.

Don't you need kernel level access to use this file?

Got any tips for people just starting out new monkeying with low level stuff.

(Oct 21, 2025 09:16 PM)enki Wrote: [\[CITE\]](#) :D happy researching!

You will need to do a good amount of work for real connections, but it's possible.

Please be very careful... low level work can be messy if you don't undo certain changes, or if you mess with your machine too much!

It's actually good to do this. However, I think there's a lot of ways to BOS the socket stack.

It requires driver level kernel access. I think you would have to use an VM if you want to test things because it can damage your computer.

However, once you do this you can make your own lib and then use that instead. It's more secure meaning you won't have to rely heavily on Microsoft code.

I don't know if this is practical thought. They said online that it's best to use a kernel level version of winsock.

The reason they said the AFD.sys depends on the machine you run it on and the windows version it will have different values which could easily brick the system if it's wrong.

I will look more into it later in a couple of days.

HiddenEyE - October 27, 2025, 6:46 pm

nice post mate. that code can be extended to be more sophisticated

Hackers - November 5, 2025, 6:23 am

(Oct 27, 2025 06:46 PM)HiddenEyE Wrote: [\[CITE\]](#) nice post mate. that code can be extended to be more sophisticated

Definitely, I would like to see some people give it a go

[ttbr0 CL1B](#) are either of you up to the task? You could push this forward for the entire internet, there isn't much public past this starting POC.

CL1B - November 5, 2025, 6:29 am

Sick, I'll check this out and hopefully follow up.

[itcorp](#) , check this out.

Hackers - November 5, 2025, 6:37 am

(Nov 5, 2025 07:29 AM)MarlboroMan Wrote: [\[CITE\]](#) Sick, I'll check this out and hopefully follow up.

[itcorp](#) , check this out.

Perfect, I figured you'd be interested in this :)

arsium - November 7, 2025, 5:24 am

Quite nice and interesting post. I will take a look for sure.

Hackers - November 7, 2025, 5:26 am

(Nov 7, 2025 06:24 AM)arsium Wrote: [\[CITE\]](#) Quite nice and interesting post. I will take a look for sure.

let us know what you find

arsium - November 7, 2025, 6:04 am

(Nov 7, 2025 06:26 AM)nodes Wrote: [\[CITE\]](#) let us know what you find

Go to mswsock.dll. Inspect WSP*Proc. WSPListen, WSPAccept...

And here is :

```
Code 180001e38          Status = NtDeviceIoControlFile(*(uint64_t*)(rax_3 + 8),
180001e38              *(uint64_t*)((char*)rdi_1 + 0x10), nullptr, nullptr,
180001e38              &IoStatusBlock, 0x1200b, &var_50, 0xc, 0, 0);
WSPListen :
```

```
Code 180001ddd          struct IO_STATUS_BLOCK IoStatusBlock;
180001ddd              *(uint32_t*)((char*)IoStatusBlock + 0x14) = (uint32_t)r14;
180001deb              uint8_t var_48_1 = *(uint8_t*)(rax_3 + 0x45) >> 2 & 1;
1800130d5              NTSTATUS Status;
1800130d5
```

```

1800130d5 do
1800130d5 {
180001dfb     IoStatusBlock.Information = SockSanEnabled;
180001e38     var_50;
180001e38     Status = NtDeviceIoControlFile(*(uint64_t*)(rax_3 + 8),
180001e38         *(uint64_t*)((char*)rdi_1 + 0x10), nullptr, nullptr,
180001e38         &IoStatusBlock, 0x1200b, &var_50, 0xc, 0, 0);
180001e38
180001e49     if (Status == STATUS_PENDING)
180001e49     {
180001e4e         LeaveCriticalSection(rax_3 + 0xe0);
180001e6b         SockWaitForSingleObject(*(uint64_t*)((char*)rdi_1 + 0x10),
180001e6b             arg1, 3, 4);
180001e73         EnterCriticalSection(rax_3 + 0xe0);
180001e7f         Status = IoStatusBlock..Status;
180001e49     }
180001e49
180001e85     if (Status >= STATUS_SUCCESS)
180001e85     {
180001e93         int32_t rax_5 = SockNotifyHelperDll(rax_3, 2);
180001e98         rdi_2 = rax_5;
180001e98
180001e9c         if (rax_5)
180001e9c             goto label_18001307b;
180001e9c
180001ea2         *(uint8_t*)(rax_3 + 0x44) |= 1;
180001ea2
1800130ed         if (Ws2SqmInitStatus == 1 && *(uint32_t*)(rax_3 + 0x20) == 1)
1800130ed         {
1800130f6             int32_t rax_10 = data_180063b4c;
1800130f6             data_180063b4c += 1;
1800130f6
180013103             if (rax_10 + 1 > 0)
180013103             {
180013109                 void* Ws2SqmInfo_1 = Ws2SqmInfo;
180013109
180013113                 if (rax_10 + 1
180013113                     > *(uint32_t*)((char*)Ws2SqmInfo_1 + 0x50))
180013119                     *(uint32_t*)((char*)Ws2SqmInfo_1 + 0x50) =
180013119                         rax_10 + 1;
180013103             }
1800130ed         }
1800130ed
180001eb7         if (*(uint8_t*)(rax_3 + 0x45) & 0x10)
180001eb9             SockTLNPIListenerCount += 1;
180001eb9
180001ec4         int32_t rax_6 = SockSetHandleContext(rax_3);
180001ec9         rdi_2 = rax_6;
180001ec9
180001ecd         if (rax_6)
180001ecd             goto label_18001307b;
180001ecd
180001ed9         if (SockSanEnabled == rax_6)

```

```

180013121          goto label_180001edf;
180013121
180013121          int32_t rax_12 = *(uint32_t*)(rax_3 + 0x4c);
180013121
180013132          if (rax_12 != data_1800626f4 && rax_12 != data_180062474)
180013132              goto label_180001edf;
180013132
18001313e          int32_t rax_13 = SockSanListen(rax_3, (uint32_t)r14);
18001313e
180013145          if (!rax_13)
180013145              goto label_180001edf;
180013145
180013152          if (!(data_180063ae0 & 1))
180013152              goto label_180001eec;
180013152
180013167          WPP_SF_D(0xb,
180013167              &WPP_95f507230b843ef815faab08b2fb8dc7_Traceguids, rax_13);
180013167          goto label_18001316c;
180001e85      }
180001e85
1800130cc          if (Status != STATUS_INVALID_PARAMETER_12)
1800130cc              break;
1800130cc
1800130ce          Status = SockSanActivate();
1800130d5      } while (Status >= STATUS_SUCCESS);
Currently, I did not dynamic analysis while debugging.

```

Remember the article for bind (WSPBind) for example the IOCTL Code ([Part2](#)):

```

0x12003 = #define IOCTL_AFD_BIND          _AFD_CONTROL_CODE(AFD_BIND,
METHOD_NEITHER) // 0x12003
Code 180004cbc          (uint32_t)var_110 = 0x12003;
180004cdf          NTSTATUS Status_1 = NtDeviceIoControlFile(*(uint64_t*)(rax_4 + 8),
180004cdf          *(uint64_t*)((char*)r15_1 + 0x10), nullptr, nullptr,
180004cdf          &IoStatusBlock, (uint32_t)var_110, rsi, InputBufferLength, rsi,
180004cdf          OutputBufferLength);

```

ttbr0 - November 8, 2025, 6:38 pm

(Nov 5, 2025 07:23 AM)nodes Wrote: [\[CITE\]](#) [ttbr0 CL1B](#) are either of you up to the task? You could push this forward for the entire internet, there isn't much public past this starting POC.

I've already seen this make its way around my space before and its first mention was a while back dating back to 2020 with just whispers and then a poc back to 2022. it isnt particularly interesting enough for me to do anything about it but i would follow any development on it.

Hackers - November 9, 2025, 4:51 am

(Nov 8, 2025 07:38 PM)ttbr0 Wrote: [\[CITE\]](#) I've already seen this make its way around my space before and its first mention was a while back dating back to 2020 with just whispers and then a poc back to 2022. it isnt particularly interesting enough for me to do anything about it but i would follow any development on it.

Fair enough, do you know if they hit a wall or had any technical reasons to not continue? Was surprised to not find much on this topic

ttbr0 - November 9, 2025, 9:11 pm

(Nov 9, 2025 05:51 AM)nodes Wrote: [\[CITE\]](#) Fair enough, do you know if they hit a wall or had any

technical reasons to not continue? Was surprised to not find much on this topic

i dont know. personally i dont really see why evasion on socket operations is useful when you can observe and track network data outside of the operating system. it might also be suspicious that a program is directly interacting with the driver. if i saw this in the wild, i would be immediately suspicious than if i saw something just using winsock functions.

Hackers - November 10, 2025, 5:13 am

(Nov 9, 2025 10:11 PM)ttbr0 Wrote: [\[CITE\]](#) i dont know. personally i dont really see why evasion on socket operations is useful when you can observe and track network data outside of the operating system. it might also be suspicious that a program is directly interacting with the driver. if i saw this in the wild, i would be immediately suspicious than if i saw something just using winsock functions. Agreed, it would be instantly suspect

My thought is maybe red teams could use it with other methods and stay undetected if the blue team is not looking for this.

ttbr0 - November 10, 2025, 10:40 pm

(Nov 10, 2025 06:13 AM)nodes Wrote: [\[CITE\]](#) Agreed, it would be instantly suspect

My thought is maybe red teams could use it with other methods and stay undetected if the blue team is not looking for this.

it could be practically viable somewhere but i dont know. i think its theoretically possible for edr to detect this with high fidelity but im not sure if its implemented in the real world.

itcorp - November 15, 2025, 6:43 am

(Oct 21, 2025 08:25 PM)nodes Wrote: [\[CITE\]](#) Low Level AFD.sys
Socket Creation
for Windows 11

I found this write-up about some undocumented functions in Windows.

This method uses AFD.sys on Win11, and does not use Winsock to open a socket.

Further research with IOCTL can be done to identify how the socket can be used.

This guide can help blue teams & cybersecurity researchers defend their machines by teaching them more about the lesser known & undocumented AFD.sys interface!

Software developers can write more efficient code, low level, and without relying on a boring library like Winsock. I'll be interested in hearing what people can discover or learn from this.

Read the Paper:

<https://github.com/LNodesL/blue-team-defense-research/blob/f043f9b5ba79aa8eaa1cce75b7ca0cc761964c12/Windows/2025-07-16%20-%20Under%20the%20Hood%20of%20AFD.sys%20-%20Investigating%20Undocumented%20Interfaces.pdf>

Mirror:

<https://mega.nz/file/AvUESQLS#6dUV8TN-cOfG9wjRPm-oCPB9nA9JY33ahyw0PdI36ak>

Here is the example code below that they gave. Keep in mind though that your machine's values may be different, and you shouldn't run this without reading the attached PDF first...

Spoiler ([Click to View](#))

```
Code [h1]include <stdint.h>[/h1]
[h1]include <Windows.h>[/h1]
[h1]include <winternl.h>[/h1]
[h1]include <iostream>[/h1]
[h1]pragma comment(lib, "ntdll.lib")[/h1]
enum AFD_ENDPOINT_FLAGS : uint32_t {
    AFD_ENDPOINT_FLAG_CONNECTIONLESS = 0x0000000000000001,
    AFD_ENDPOINT_FLAG_MESSAGE_MODE = 0x0000000000000010,
    AFD_ENDPOINT_FLAG_RAW = 0x0000000000000100,
    AFD_ENDPOINT_FLAG_MULTIPOINT = 0x0000000000001000,
    AFD_ENDPOINT_FLAG_CROOT = 0x0000000100000000,
    AFD_ENDPOINT_FLAG_DROOT = 0x0000010000000000,
    AFD_ENDPOINT_FLAG_IGNORETDI = 0x0001000000000000,
    AFD_ENDPOINT_FLAG_RIOCKET = 0x0100000000000000,
};
struct AFD_OPEN_PACKET_EA {
    uint32_t nextEntryOffset;
    uint8_t flags;
    uint8_t eaNameLength;
    uint16_t eaValueLength;
    char eaName[0x10];
    uint32_t endpointFlags;
    uint32_t groupID;
    uint32_t addressFamily;
    uint32_t socketType;
    uint32_t protocol;
    uint32_t sizeOfTransportName;
    uint8_t unknownBytes[0x9];
};
NTSTATUS createAfdSocket(PHANDLE socket) {
    const char* eaName = "AfdOpenPacketXX";
    UNICODE_STRING devName;
    RtlInitUnicodeString(&devName, L"\\Device\\Afd\\Endpoint");

    OBJECT_ATTRIBUTES object;
    object = { 0 };
    object.ObjectName = &devName;
    object.Length = 48;
    object.Attributes = 0x40;
    AFD_OPEN_PACKET_EA afdOpenPacketEA;
    afdOpenPacketEA.nextEntryOffset = 0x00;
    afdOpenPacketEA.flags = 0x00;
    afdOpenPacketEA.eaNameLength = 0x0F;
    afdOpenPacketEA.eaValueLength = 0x1e;
    afdOpenPacketEA.endpointFlags = 0x00;
    afdOpenPacketEA.groupID = 0x00;
    afdOpenPacketEA.addressFamily = AF_INET;
    afdOpenPacketEA.socketType = SOCK_STREAM;
    afdOpenPacketEA.protocol = IPPROTO_TCP;
```

```

afdOpenPacketEA.sizeOfTransportName = 0x00;
memset(afdOpenPacketEA.eaName, 0x00, 0x10);
memcpy(afdOpenPacketEA.eaName, eaName, 0x10);
memset(afdOpenPacketEA.unknownBytes, 0xFF, 0x9);
IO_STATUS_BLOCK IoStatusBlock;
return NtCreateFile(socket, GENERIC_READ | GENERIC_WRITE | SYNCHRONIZE, &object,
&IoStatusBlock, 0, 0, FILE_SHARE_READ | FILE_SHARE_WRITE,
FILE_OPEN_IF,
FILE_SYNCHRONOUS_IO_NONALERT, &afdOpenPacketEA,
sizeof(afdOpenPacketEA));
}
int main() {
HANDLE socket;
NTSTATUS status = createAfdSocket(&socket);
if (!NT_SUCCESS(status)) {
std::cout << "[-] Could not create socket: " << std::hex << status <<
std::endl;
return 1;
}
std::cout << "[+] Socket created!" << std::endl;
return 0;
}
Read and learn,
Be a legend.

```

really really interesting exactly,nice share as always bud

Hackers - December 7, 2025, 10:01 am

Here is a version created in Go/lang that can be compiled for Windows, even from a MacOS build machine:

<https://github.com/LNodesL/Golang-AFD.sys-Socket-Creation>

LiNixx - January 14, 2026, 12:18 am

This is a fascinating subject thank you for the share.

I've successfully ported your code to delphi and can open a socket but I am stuck creating packets

Frustrating but fun in the same time lol

Windows 10, version 22H2, fully patched up to build 6456 vm I dont have any other vms spun up at the moment.

Delphi 12.1

If nothing else its been a time consumer lol :P

Here's what I have if anyone is interested, gotta stop need sleep

Code <https://pastebin.com/Qnzkawaa>

Phantom_NiNja - January 16, 2026, 10:41 pm

(Jan 14, 2026 01:18 AM)LiNixx Wrote: [\[CITE\]](#) This is a fascinating subject thank you for the share.

I've successfully ported your code to delphi and can open a socket but I am stuck creating packets

Frustrating but fun in the same time lol

Windows 10, version 22H2, fully patched up to build 6456 vm I dont have any other vms spun up at the moment.

Delphi 12.1

If nothing else its been a time consumer lol :P

Here's what I have if anyone is interested, gotta stop need sleep

Code <https://pastebin.com/Qnzkawaa>

You should know this MR elite software engineer. Maybe go on github or use ai to help you.

No need to code anything yourself right? lol

Hackers - January 17, 2026, 3:24 pm

(Jan 14, 2026 01:18 AM)LiNixx Wrote: [\[CITE\]](#) This is a fascinating subject thank you for the share.

I've successfully ported your code to delphi and can open a socket but I am stuck creating packets

Frustrating but fun in the same time lol

Windows 10, version 22H2, fully patched up to build 6456 vm I dont have any other vms spun up at the moment.

Delphi 12.1

If nothing else its been a time consumer lol :P

Here's what I have if anyone is interested, gotta stop need sleep

Code <https://pastebin.com/Qnzkawaa>

ahhh maybe I'll go back to my Golang one too and try, it was fun and time consuming like you said lol. I need to test a few more things but think I figured it out a few weeks ago