

# GameNetworkingSockets: gamertime

Krish Magal

Any modern video game has some online aspect to it, whether it's online multiplayer, in game shop, or cloud saves. Developers can spend a large amount of time trying to find good libraries to use for their games, and some are difficult to use without additional libraries, further complicating the process.

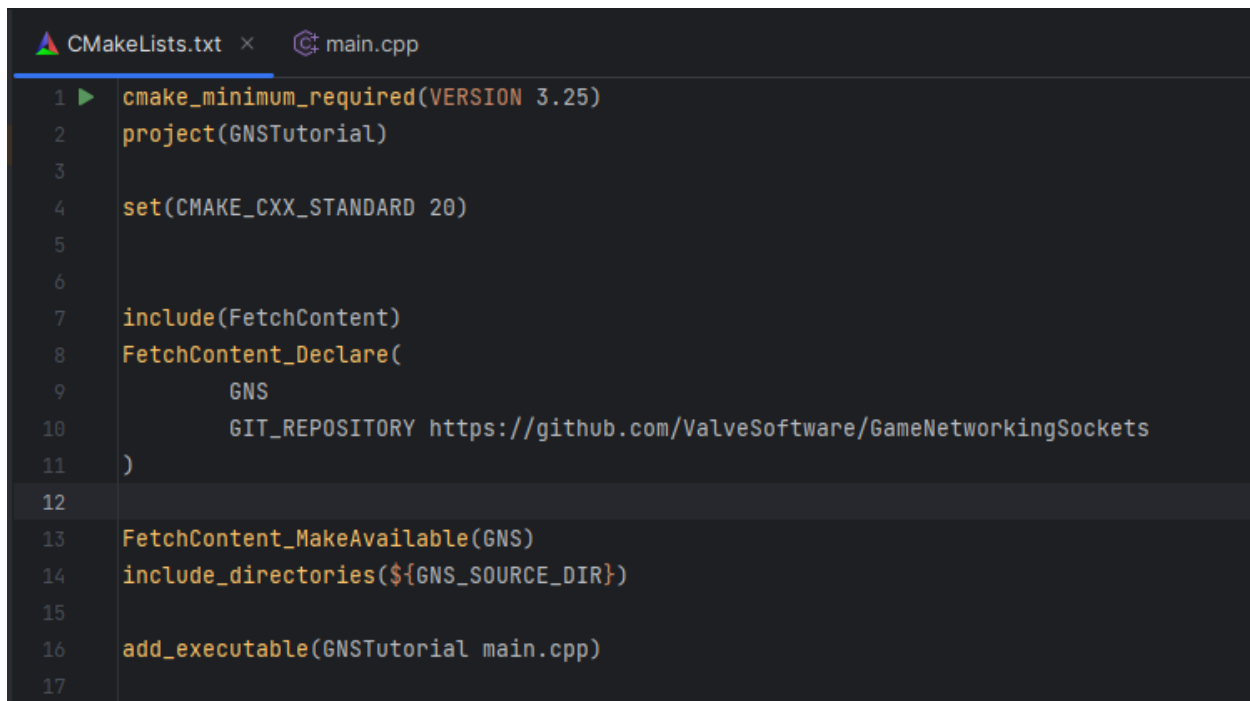
One library that makes a strong case to be used in the development of video games is Valve Software's GameNetworkingSockets (GNS) library. This library is one created by one of the largest names in video games, Valve Software, best known for the creation of Steam and its following products. This library is very well made. It's preferred by many studios trying to publish games on Steam.

"The main interface class is named SteamNetworkingSockets, and many files have "steam" in their name. But *Steam is not needed*. If you don't make games or aren't on Steam, feel free to use this code for whatever purpose you want."

Without further ado, let's pull back the curtain and see what we can find. When getting started, I found the easiest thing for me to do was to set up a simple C++ project. One great thing about this library is that there are multiple wrappers available, so that you can code in either C# or Python, showing just how accessible this library is!

For this, I set up a simple CMake project. I found it was the easiest way that I could get up and running, since I've got a decent amount of experience with it. Of course, you can set up your configuration any way you like, so mileage may vary. Below is an example of what your CMakeLists.txt might look like just to get started, we could be updating

this.

A screenshot of a code editor with a dark theme. The editor has two tabs at the top: 'CMakeLists.txt' (active) and 'main.cpp'. The 'CMakeLists.txt' tab shows a CMake script with the following content:

```
1 cmake_minimum_required(VERSION 3.25)
2 project(GNSTutorial)
3
4 set(CMAKE_CXX_STANDARD 20)
5
6
7 include(FetchContent)
8 FetchContent_Declare(
9     GNS
10     GIT_REPOSITORY https://github.com/ValveSoftware/GameNetworkingSockets
11 )
12
13 FetchContent_MakeAvailable(GNS)
14 include_directories(${GNS_SOURCE_DIR})
15
16 add_executable(GNSTutorial main.cpp)
17
```

As you can see, it's very simple to be able to fetch the library from GitHub, no installation necessary! If you're interested in checking out the GitHub repo, I've linked it [here](#). For this project, I've set up a simple P2P messaging system. P2P games are great for LAN parties, and games that don't need a true internet connection.

Here's a simple main function that creates 2 players and binds them to 2 sockets. Once the sockets have been bound, the client begins polling for incoming messages.

```

int main() {
    if (!InitializeGameNetworkingSockets()) {
        // Handle initialization failure
        return 1;
    }

    // Set up identity for both players
    GameNetworkingIdentity player1Identity;
    GameNetworkingIdentity player2Identity;

    // Create a listen socket for player 1
    HSteamListenSocket player1ListenSocket = GameNetworkingSockets_CreateListenSocketP2P(player1Identity, 0, 2, NULL, 0);

    // Create a P2P connection from player 2 to player 1
    HSteamNetConnection player2Connection = GameNetworkingSockets_ConnectP2P(player2Identity, player1Identity, NULL, 0);

    // Wait for messages
    while (true) {
        SteamNetworkingMessage_t* incomingMsg;
        int numMessages = GameNetworkingSockets_PollIncomingMessages();

        for (int i = 0; i < numMessages; ++i) {
            if (GameNetworkingSockets_ReceiveMessagesOnConnection(incomingMsg, 1, NULL) > 0) {
                // Handle incoming messages
                OnIncomingMessage( callback: incomingMsg);
            }
        }
    }

    // Shutdown and cleanup
    GameNetworkingSockets_Kill();
    return 0;
}

```

Here's what we might do when we get a message:

```

#include <steam/steamnetworkingsockets.h>

// Function to initialize GameNetworkingSockets
bool InitializeGameNetworkingSockets() {
    return GameNetworkingSockets_Init();
}

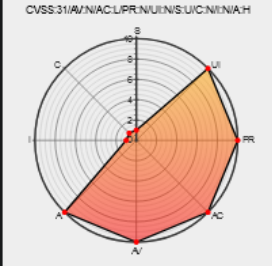
// Callback to handle incoming messages
void OnIncomingMessage(SteamNetConnectionStatusChangedCallback_t* callback) {
    int msgSize;
    int bytesRead = GameNetworkingSockets_ReceiveP2PPacket(callback->m_hConn, NULL, &msgSize);

    if (bytesRead > 0) {
        char* buffer = new char[msgSize];
        bytesRead = GameNetworkingSockets_ReceiveP2PPacket(callback->m_hConn, buffer, &msgSize);
        if (bytesRead > 0) {
            printf("Received message: %s\n", buffer);
        }
        delete[] buffer;
    }
}

```

Unfortunately, there are still some bugs to be worked out before we can get some output, GNS requires the use of another library, Protocol Buffer a.k.a. Protobuf for serialization. I'm still working out the bugs on how to include it with my project, so stay tuned! I hope that I've got something to show you all in the next day or two.

In the meantime, here's the most recent CVE report on the vulnerability score of the GNS library.



Certain versions of **Game Networking Sockets** from **Valvesoftware** contain the following vulnerability:

Valve's Game Networking Sockets prior to version v1.2.0 improperly handles inlined statistics messages in function `CConnectionTransportUDPBase::Received_Data()`, leading to an exception thrown from libprotobuf and resulting in a crash.

It currently features a score of 7.5, a high vulnerability score. Let's break it down.

CVSS3 Score: 7.5 - HIGH			
Attack Vector <sup>Ⓢ</sup>	Attack Complexity	Privileges Required	User Interaction
NETWORK	LOW	NONE	NONE
Scope	Confidentiality Impact	Integrity Impact	Availability Impact
UNCHANGED	NONE	NONE	HIGH

As you can see here, this flaw features a low attack complexity and a high availability impact. Developers can rest easy however, as it has neither any confidentiality impact nor any integrity impact. Unfortunately, it's not made clear if this is the proprietary version, or rather the open-source version. I would hope that some brave network engineer found a way to fix this.