Client>Server Communication

There is no singular function that is responsible for this task. In Apex, we rather have multiple of those working together.

Let's get started with the most common function, **R5AC::PushViolation:**

```
void __fastcall R5AC::PushViolation(
        const __m128i *pszIdentifierStr,
        char Severity,
        __int64 ExtraDataPtr,
        __int64 ExtraDataLen);
```

In case of client-side anomalies being detected, it will be invoked like this:

```
LBL_ON_ABNORMALITY_FOUND:
            v119 = v116();
            v120 = 0;
            v121 = 0i64;
            do
            {
              v122 = *(_BYTE *)(v121 + 41489128); // these are just data and key, and the RVA can be computed with analysis of a runtime dump.
              ++v120;
              v123 = *(_BYTE *)(v121 + 27928337);
              v188[++v121 + 15] = v123 ^ v122;
            }
            while ( v120 < 0xC ); // C-String encryption using simple xor
            sprintf_s_2(Buffer, 0x104ui64, Format, v119);
          }
        }
        R5AC::PushViolation(v200, 1, v177, (__int64)Buffer);
```

It's purpose is to enqueue a new violation record into a list that is fetched and emptied by numerous callers, all originating from **r5apex_dx12.exe.** These are to be considered slaves because they simply check if there is any messages pending, use CLC_AntiCheat virtual method

table to setup a new netmessage in source engine, and then puts the data of the anti-cheat violation/message into the netmsg and sends it to the game server.

```
if ( gpNetChan )
    {
      v14 = vft::CLC_AntiCheatMsg;
      v15 = 0;
      v17 = 0i64;
      v5 = 5;
      v16 = 1;
      // send all of them in bulk
      for ( i = 0; (unsigned __int8)R5AC::PopAnticheatMsg(v20, &i); --v5 )
      {
        if ( !v5 )
          break;
        v18 = v20;
        v19 = i;
        C_NetChan::SendNetMsg(v4, &v14, 0, 0);
        C_NetChan::SendDatagram(v4, 0i64);
      }
    }
```

Additionally, there is a couple more ways for the anti-cheat to phone back home. Let me introduce you **R5AC::TrySendViolationOrQueue.** It behaves almost identically to it's predecessor, with the twist that if your game client has an active network channel instance (or in other words: an active connection to a game server is present), it will bypass it's usual queue-based system and directly send it to the game server.

```
void __fastcall R5AC::TrySendViolationOrQueue(
        const __m128i *id_str,
        __int64 a2,
        __int64 extraData,
        const __m128i *extraDataLen)
  {
    --*(_DWORD *)(v4 + 68);
    v5 = gpNetChan;
    if ( gpNetChan && (unsigned int)MEMORY[0x7FFE8D2B5550](id_str, 0i64) ==
  dword_594427C )
    {
      v28 = 0;
      v9 = id_str;
```

```
    do
    {
      v10 = v9->m128i_i8[0];
      v9->m128i_i8[pszBuffer - (char *)id_str] = v9->m128i_i8[0];
      v9 = (const __m128i *)((char *)v9 + 1);
    }
    while ( v10 );

    idStrLength = -1i64;
    do
      ++idStrLength;
    while ( id_str->m128i_i8[idStrLength] );
    v12 = &pszBuffer[idStrLength + 1];
    if ( extraDataLen )
    {
      idx = -1i64;
      do
        ++idx;
      while ( extraDataLen->m128i_i8[idx] );
      R5::AllocAndCopyStruWithHdr(
        &mem,
        extraDataLen,
        idx,
        (__int64 (__fastcall **)(_QWORD, unsigned __int64, __int64))off_1AA6728);
      extraDataPtr = mem;
    }
    else
    {
      extraDataPtr = 0i64;
      mem = 0i64;
    }
    *(_DWORD *)v12 = 1;
    v12[4] = 0;
    *(_QWORD *)(v12 + 5) = extraData;
    sse_memcpy((__m128i *)(v12 + 13), extraDataPtr, (unsigned
int)(extraDataPtr[-1].m128i_i32[3] + 1));
    v15 = (_DWORD)v12 - ((unsigned int)&v27 + 880) + 14 + extraDataPtr[-
1].m128i_i32[3];
    someKey = 0x42A1110F96B5E116i64;
    v16 = 0;
    if ( v15 != 2 )
    {
      v17 = pszBuffer;
      do
      {
```

```
        ++v17;
        v18 = v16++ & 7;
        *(v17 - 1) ^= *((_BYTE *)&someKey + v18);
       }
       while ( v16 < v15 - 2 );
      }
      v26 = v15;
      *(_QWORD *)&v21 = vft::CLC_AntiCheatMsg;
      v22 = 0;
      v25 = &v28;
      v24 = 0i64;
      v23 = 1;
      C_NetChan::SendNetMsg(v5, &v21, 0, 0);
      C_NetChan::SendDatagram(v5, 0i64);
      sse_memset((__int64)&v28, 0, 0x400ui64);
      *(_QWORD *)&v21 = &off_173A878;
      if ( _InterlockedExchangeAdd(&extraDataPtr[-1].m128i_i32[2], 0xFFFFFFFF)
    == 1 )
        (*(void (__fastcall **)(__int64))(extraDataPtr[-1].m128i_i64[0] +
    8))(extraDataPtr[-1].m128i_i64[0]);
     }
     else
     {
      R5AC::PushViolation(id_str, 0, extraData, (__int64)extraDataLen);
     }
    }
```

“

Revision #59
Created 2026-02-12 22:54:45 UTC by Admin
Updated 2026-02-17 06:37:47 UTC by Admin