Season 30

Type	captured at	game build	memory dump completeness	links	Comment
Generic	16.10.2026	R5pc_r5-301_J65_CL11630615_FSv30_1_2026_10_02_09_54	N/A	**No IDB Present**	

> **AC Migration:** The game is now protected by EA Javelin instead of EAC. Alongside it, you will find R5AC still being deployed and doing similar stuff that we uncovered back in the day. Additionally, client-side code is present for an integration of Anybrain, a solution advertised as an AI-based cheat detection. Although it should be mentioned that it wasn't active within my own limited tests.

> **Packer Removal (partly):** It seems like respawn is no longer applying dynamic memory protection onto the primary game executable, r5apex_dx12.exe. I'm unable to claim that everything related to their old Theia implementation is gone, as it was doing much more than just packing, for example it also hard stuff for hardware identification and broader anti-tamper functionality. This means you will be able to obtain a full memory dump without having to defeat a packer that periodically re-encodes stale pages.

Useful Codesites and Globals

Experiments System

Name	Address	Comment

R5::FetchExperimentsForProfile	%DUMP_BASE%+0x1AC7A10	Sends an HTTP/S request towards an endpoint URL "experimentation/v1/fetch-experiment-grouping". It attaches various parameters including Player/Nucleus-ID, and then receives a list of experiments and their individual states.
R5::ParseExperiments	%DUMP_BASE%+0x1AC8430	Parses the experiment list that was previously received with the previous function. It has a special handling case for "8-1-anyb-rollout", which if the experiment is determined as active and you're within the active treatment group, a ConVar named raw_input_client_send_enable is enabled.

Input Telemetry (Anybrain)

R5::ProcessInputTelemetry_Opaque_1	%DUMP_BASE%+0x1CBDD60	Processes some input telemetry related data, and conditionally writes it to a global ring buffer.
R5::ProcessInputTelemetry_Opaque_2	%DUMP_BASE%+0x1D830B0	
R5::InputTelemetrySender::1	%DUMP_BASE%+0x1D834D0	Used to send player input telemetry using standard CNetChan facilities. (SendNetMsg+SendDatagram)
R5::InputTelemetrySender::2	%DUMP_BASE%+0x1D82E00	Uses sub_1A36B90 to indirectly invoke R5::InputTelemetryDispatcher::1
R5::InputTelemetryDispatcher::1	%DUMP_BASE%+0x1D832A0	Used to send player input telemetry using standard CNetChan facilities. (SendNetMsg+SendDatagram)

Game Engine

CL_Move	%DUMP_BASE%+0x1AF4C30	
CL_SendMove	%DUMP_BASE%+0x1AF49E0	
C_ClientRenderable::SetupBones	%DUMP_BASE%+0x1FE6950	

R5AC

EnqueueViolation	%DUMP_BASE%+0x1D85A90	Appends a new anti-cheat telemetry message into the global queue, in many cases for a violation of some sort, but it hasn't been proved that it is only used for violations.
EnqueueOrSendViolationDirectly	%DUMP_BASE%+0x1D85BF0	Essentially something similar as the above EnqueueViolation function, with one difference being that it has a fast path: If your game client has an active network channel to a game server, it will attempt to send the anti-cheat telemetry message directly using engine API's like C_NetChan::SendNetMsg and C_NetChan::SendDatagram.). Otherwise it just calls into EnqueueViolation to do the usual queueing.
InitializeStackWalker	%DUMP_BASE%+0x1D83E40	Responsible for various things: • Initialization of imports for R5AC stack walker / control flow inspection • Initialization of the code execution whitelist. Its exact format is to be studied. • Sets a global variable of type u8 to 1 when initialization succeeds. The inlined stack walker code in game & engine functions then checks this alongside their other calculations for determining whether the stack walker should run. My understanding of this list is that it also is very likely a region-based whitelist and not per module, like you would initially think. So they kept that design choice and just switched it with a hash map. There seems to be some other changes too, including how the whitelist is built. In one occasion, it searches for a specific module in the LDR, loops through its regions and adds any section that is executable, non-discardable to its whitelist structure.

Useful Patches

Below is a list of convenient patches that you can apply, to enhance your experience within Apex. The best part is you're not introducing anything artifical into the mix. If you're a controller player and already used to the feeling of aim assist, for example, you can just buff it directly. Note that this is probably no longer a golden ticket with the rise of Artifical Intelligence in EA's playbook.

Another positive aspect is that you don't necessarily need to patch anything to achieve much of these. For example, with the name hider, you can just manipulate a couple of global variables within Apex to make the lookup fail. For the aim assist stuff, you could reverse engineer their playlist variable system more deeply, and then figure out how to do it more stealthily.

Same thing applies for the deadzone check. Basically, inside a mouse related processing function at r5apex_dx12.exe, it just sets some global u64 which represents the last time your mouse has moved. You can log the values, see when they change, improve your model of the surrounding code and then find a way to achieve it in a much more subtle way. My approach saved me a lot of time, because for the most time I could afford changing whatever I wanted. This is subject to change but even if that's no longer sustainable, it's still valuable for things like narrowing down the amount of code you need to reverse engineer.

Patch Name	Address / Offset	Patch Bytes
Gamepad Assist Strength Changer	`0x20A83F0`	`48 31 C0 0F 57 C0 48 C7 C0 CD CC 4C 3F 66 0F 6E C0 C3` `moves 0.8f in hexadecimal representation into XMM0`
Mouse Inactivity Check	`0x20A7022`	`90 90 90 90 90 90` `just a bunchof NOPs`
Deadzone Check	`0x20A8753`	`EB`
Should Enable Aim Assist	`0x20A6FE0`	`B0 01 C3`
Input Enforcer 1	`0x21B1380`	`B0 01 C3`
Input Enforcer 2	`0x21B12E0`	`B0 01 C3` `when patching this and the first one, aim assist will work on mouse and keyboard. a controller needs to be attached nonetheless even if you can just play on MnK afterwards.`
Name Hider	`0x1A40391`	`E9 B2 00 00 00 90` `when patched, player names in your hud and world text will appear as Unknown:Unknown`

> EA's kernel driver is currently performing reads on various addresses within the game's code section. Whether that is for prevention or detection, i can't tell. But it would make sense for both use cases. During a short test session, it was most prominently a couple routines inside eaanticheat.sys accessing my target pages. Once I'm able to answer some of my open questions regarding this mechanism, I will update this book to reflect any new insights. You should expect them to have multiple integrity checks for something this obvious, but let's not overstate it too early.

Revision #23
Created 2026-10-29 05:24:40 UTC by Admin
Updated 2026-11-13 11:58:42 UTC by Admin